A-50

Proceedings of the IASTED International Conference
Parallel and Distributed Computing and Systems (PDCS '99)
November 3-6, 1999, Cambridge, Massachusetts - USA

Multimessage Multicasting with Bounded Number of Destinations

Teofilo F. Gonzalez

teo@cs.ucsb.edu
Department of Computer Science
University of California
Santa Barbara, CA 93106

Abstract

We consider Multimessage Multicasting over the n processor complete static network when forwarding of messages is allowed. We present an efficient algorithm to generate a communication schedule with total communication time at most $\lfloor(2-\frac{1}{l})d\rfloor+1$, for problem instances where each processor needs to send messages to at most ld destinations for some $2 \leq l \leq d$, and d is the maximum number of messages that each processor may send (or receive). Our algorithm consists of two phases. First a set of communications are scheduled to be carried out in such a way that the resulting problem is a multimessage unicasting problem of degree d. Then we generate a communication schedule for the resulting problem instance. We also discuss multimessage multicasting for dynamic networks.

Key Words: Approximation Algorithms, Multimessage Multicasting, Dynamic Networks, Forwarding, Iterative Methods, Communication Schedules.

1 Introduction

The Multimessage Multicasting problem over the n processor complete (or fully connected) static network, MM_C, consists of constructing a communication schedule with least total communication time for multicasting (transmitting) any given set of messages. There are n processors, $P=\{P_1,P_2,\ldots,P_n\}$, executing processes, and these processes are exchanging messages that must be routed through the links of the network. Our objective is to determine when each of these messages is to be transmitted so that all the communications can be carried in the least total amount of time. *Forwarding*, which means that messages may be sent through indirect paths even though a single link paths exist, allows communication schedules with significantly smaller total communication time. This version of the multicasting problem is referred to as the MMF_C problem, and the objective is to determine when each of these messages is to be transmitted so that all the communications can be carried in the least total amount of time. We define the $l-MMF_C$, for $2 \leq l \leq d$, as the MMF_C in which each processor has at most ld edges emanating from it. Our introduction is a condensed version of the one in [6] which includes a complete justification for the multimessage multicasting problem as well as motivations, applications, and examples.

Routing in the complete static network is the simplest and most flexible when compared to other static and dynamic networks. Dynamic networks (or multistage interconnection networks) that can realize all Permutations (each in one communication phase) and Replicate data (e.g., n by n Benes network based on 2 by 2 switches that can also act as data replicators) will be referred to as *pr-dynamic networks*. Multimessage Multicasting for pr-dynamic and complete networks is not too different, in the sense that any communication schedule for a complete network can be translated automatically into an equivalent communication schedule for any pr-dynamic network. This is accomplished by translating each communication phase for the complete network into no more than two communication phases for pr-dynamic networks. The first phase replicates data and transmits it to other processors, and the second phase distributes data to the appropriate processors [11, 14]. The IBM GF11 machine [1], and the Meiko CS-2 machine use Benes networks for processor interconnection. The two stage translation process can also be used in the Meiko CS-2 computer system, and any multimessage multicasting schedule can be realized by using basic synchronization primitives. This two step translation process can be reduced to one step by increasing the number of network switches by about 50% [11, 14]. In what follows we concentrate on the multimessage multicasting problem over complete networks because it has a simple structure, and, as mentioned above, results for this network can be easily translated to pr-dynamic networks.

Let us formally define our problem. Each processor P_i *holds* the set of messages h_i and *needs* to receive the set of messages n_i. We assume that $\bigcup h_i = \bigcup n_i$, and

that each message is initially in exactly one set h_i. We define the *degree* of a problem instance as $d = \mathbf{max}\{|\ h_i\ |, |\ n_i\ |\}$, i.e., the maximum number of messages that any processor sends or receives. Consider the following example.

Example 1. Processors P_1, P_2, and P_3 send messages, and the remaining six processors receive messages [1]. The hold vector is: $h_1 = \{a, b\}, h_2 = \{c, d\}, h_3 = \{e, f\}, h_4 = h_5 = h_6 = h_7 = h_8 = h_9 = \emptyset$, and the need vector is: $n_1 = n_2 = n_3 = \emptyset$, $n_4 = \{a, c, e\}, n_5 = \{a, d, f\}, n_6 = \{b, c, e\}, n_7 = \{b, d, f\}, n_8 = \{c, d, e\}, n_9 = \{c, d, f\}$. The density d is 3, and $n = 9$.

One may visualize problem instances by directed multigraphs. Each processor P_i is represented by the vertex labeled i, and there is a directed edge (or branch) from vertex i to vertex j for each message that processor P_i needs to transmit to processor P_j. The set of directed edges or branches associated with each message are *bundled* together. The problem instance given in Example 1 is depicted in Figure 1 as a directed multigraph with additional thick lines that identify all edges or branches in each bundle.

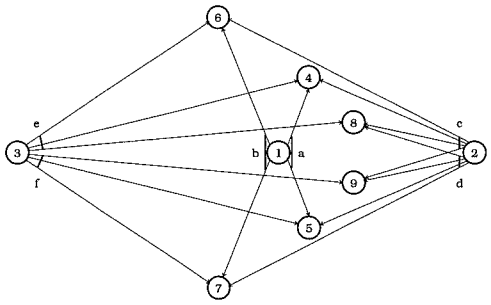

Figure 1: Directed Multigraph Representation for Example 1.

The communications allowed in our complete network must satisfy the following two restrictions.

1.- During each time unit each processor P_i may transmit one of the messages in its hold set h_i to any subset of processors. The message also remains in its hold set.

2.- During each time unit each processor may receive at most one message. The message that processor P_i receives (if any) is added to its hold set at the end of the time unit.

The communication process ends when each processor holds all the messages it needs, i.e., $n_i \subseteq h_i$. Our communication model allows us to transmit any of the messages in one or more stages. This reduces the total communication time and further reductions are possible by allowing forwarding [5].

[1] Note that in general processors may send and receive messages.

A *communication mode* C is a set of tuples of the form (m, l, D), where l is a processor index ($1 \leq l \leq n$), and message $m \in h_l$ is to be multicasted from processor P_l to the set of processors with indices in D. In addition the set of tuples in a communication mode C must obey the following communications rules imposed by our network:

1.- All the indices l in C are distinct, i.e., each processor sends at most one message; and

2.- Every pair of D sets in C are disjoint, i.e., every processor receives at most one message.

A *communication schedule* S for a problem instance I is a sequence of communication modes such that after performing all of these communications every processor holds all the messages it needs. The *total communication time* is the number of communication modes in schedule S, which is identical to the latest time there is a communication. Our problem consists of constructing a communication schedule with least total communication time. From the communication rules we know that every degree d problem instance has at least one processor that requires d time units to send, and/or receive all its messages. Therefore, d is a trivial lower bound for the total communication time.

1.1 Previous Work

The *basic multicasting problem*, BM_C, consists of all the degree $d = 1$ MM_C problem instances, and can be trivially solved by sending all the messages at time zero. When the processors are connected via a pr-dynamic network a communication mode can be performed in two stages: the data replication step followed by the data distribution step [11, 14]. This two stage process can be used in the MEIKO CS-2 machine [6]. An important subproblem is when every message it to be sent to adjacent numbered processors. This restricted multicast operation can be performed in one step in pr-dynamic networks [?], and in the MEIKO CS-2 machine. This is important to note since all the multicasting operations performed by our algorithms have this characteristic.

Gonzalez [6] also considered the case when each message has fixed *fan-out* k (maximum number of processors that may receive a given message). When $k = 1$ (multimessage unicasting problem MU_C), the problem reduces to coloring the edges of a directed bipartite multigraph so that no two edges incident upon the same vertex, and not two edges incident from the same vertex are assigned the same color. It is well known that this problem can be solved in polynomial time and that it can be colored with d colors, where d is the degree of the graph. Currently, the fastest way to solve this problem is to reduce it to the Makespan Openshop Preemptive Scheduling problem [8], which is a generalization of

this multigraph coloring problem. Every degree d multimessage unicasting problem instance has a communication schedule with total communication time equal to d. The interesting point is that each communication mode translates into a single communication step for processors interconnected via permutation networks (e.g., Benes Network, Meiko CS-2, etc.), because in these networks all possible one-to-one communications can be performed in a single communication step.

It is not surprising that several authors have studied the MU_C problem as well as several interesting variations for which NP-completeness has been established, subproblems have been shown to be polynomially solvable, and approximation algorithms and heuristics have been developed [2, 3, 10, 9, 12]. With the exception of the work in [4, 5, 6, 7, 13], research has been limited to unicasting, and all known results about multicasting are limited to single messages. Shen [13] studied multimessage multicasting for hypercube connected processors. Since hypercubes are fixed static networks, there is no direct comparison to our work. The MM_C problem involves multicasting of any number of messages, and its communication model is similar in nature to the one in the Meiko CS-2 machine, after solving some basic synchronization problems.

The MM_C problem is significantly harder than the MU_C. Gonzalez [6] showed that even when $k = 2$ the decision version of the MM_C problem is NP-complete. Gonzalez [4] developed an efficient algorithm to construct for any degree d problem instance a communication schedule with total communication time at most d^2, and presented problem instances for which this upper bound on the communication time is best possible. The lower bound holds when there is a huge number of processors and the fan-out is also huge. Since this situation is not likely to arise in the near future, Gonzalez [4, 6, 5] developed several fast approximation algorithms for problems instances with any arbitrary degree d, but small fan-out.

It is simple to show that the NP-completeness reduction for the MM_C problem given in [6] can be easily modified to establish the NP-completeness for the MMF_C problem. All the approximation results for the MM_C problem also hold for the MMF_C problem. However, Gonzalez [5] developed an efficient algorithm that generates schedules with total communication time at most $2d$ for the MMF_C problem.

In this paper we present an efficient algorithm to construct for every degree d problem instance a communication schedule with total communication time at most $\lfloor(2 - \frac{1}{l})d\rfloor + 1$ for the $l - MMF_C$ problem, where d is the maximum number of messages that each processor may send (or receive). Note that when $l = 2$ the approximation algorithm is about 25% better than the one for the algorithm given in [5].

2 Approximation Algorithm

We present in this section our algorithm to generate a communication schedule with total communication time at most $\lfloor(2 - \frac{1}{l})d\rfloor + 1$ for the $l - MMF_C$ problem, for $2 \leq l \leq d$. Our procedure consists of two steps. The first step is procedure l-FORWARD, where only those processors that initially had more than d edges may forward messages, and a subset of processors, including those with at most $d+1$ outgoing edges, will receive messages to be forwarded. After this forwarding operation we have reduced our problem to a multimessage unicasting problem which is reduced to the openshop problem and then solved by a well-known algorithm that generates a schedule with total communication time at most d. The forwarding portion has a total communication time at most $d - \lfloor\frac{d}{l}\rfloor + 1$, and therefore the resulting schedule has total communication time at most $\lfloor(2 - \frac{1}{l})d\rfloor + 1$. It is important to point out we do not allow processors to forward only one edge, because the forwarding would be impossible within the above communication time bound when we have d of these processors and only one processor receiving all the forwarded messages.

Procedure l-FORWARD first figures out the number of edges to be forwarded by each processors $\hat{F}_i$ and the maximum number of edges that can be forwarded to it r_i. This computation is carried out as follows. The first step is to transform the problem so that if the number of bundles emanating out of each processor is less than d, then all the bundles have at most one edge. This transformation is performed by transforming an edge of a multi-edge bundles into a single-edge bundle whenever the processor has fewer than d bundles and it has at least one multi-edge bundle. We define G_i as the number of edges emanating out of P_i. Now we define a tentative number of edges to be forwarded by each processor, F_i, as zero if $G_i \leq d$; 2 if G_i is $d+1$; and $G_i - d$ otherwise. Then we traverse the bundles emanating our of each processor in nondecreasing order order with respect to the number of edges in it and mark the first F_i edges that one encounters. Then we mark all the bundles with at least one marked edge. The total number of edges to be forwarder from P_i is the total number of edges in marked bundles emanating out of P_i. Then r_i is defined as $d - (G_i - \hat{F}_i)$. This guarantees that in the resulting problem at most d single-edges will emanate out of each processor.

Procedure l-FORWARD figures next the time $t(i)$ when each bundle B_i is to be forwarded. It also computes the set of processors that will receive each forwarded message. The last step is to split every multi-edge bundle into single-edge bundles. Later on we show that the transformation generates an instance of the multimessage unicasting problem of degree d.

Procedure l-**FORWARD** (I, G)
/* Remember that $2 \leq l \leq d$ */
for i=1 **to** n **do**
while P_i has fewer than d bundles and at least one multi-edge bundle **do**
delete an edge from a multi-edge bundle and add it as a single-edge bundle;
endwhile
endfor
Let G_i be the number of edges leaving P_i in G;
Tentative number of edges forwarded from P_i is

$$F_i = \begin{cases} 0 & \text{if } G_i \leq d \\ 2 & \text{if } G_i = d+1 \\ G_i - d & \text{if } G_i \geq d+2 \end{cases}$$

Traverse the bundles emanating out of P_i in non-decreasing order with respect to the number of edges in the bundle, and visit all the edges in each bundle in any order.
Mark the first F_i edges emanating out of P_i visited during the above traversal, and also mark all the bundles emanating out of P_i with at least one marked edge.
Now traverse all the marked bundles and visit ALL their edges in the same order they were visited by the above traversal. The bundles visited are labeled B_1, B_2, ..., and the edges are labeled e_1, e_2, ...
The number of edges to be forwarded from P_i is
$\hat{F}_i$ = the total number of edges in marked bundles emanating out of P_i;
Define the function $t(i)$ as
$(i-1) \bmod (d - \lfloor \frac{d}{l} \rfloor + 1) + 1$;
/* The message associated with bundle B_i will be forwarded at time $t(i)$. */
Let $r_i = d - (G_i - \hat{F}_i)$, the maximum number of edges that may be forwarded to P_i;
Define $R_i = \sum_{j=1}^{i-1} r_j$;
for i=1 **to** n **do**
if $R_{i-1} \neq R_i$ **then** define $g(h) = i$ for each edge labeled e_h, and $R_{i-1} + 1 \leq h \leq R_i$;
endfor
/* Edge e_i will be forwarded to processor $g(i)$ */
$(\hat{I}, \hat{G}) \leftarrow (I, G)$ minus all the edges in marked bundles of G;
/* The edges in $\hat{G}$ will be added to reflect the forwarding operation. */
for every processor P_j in G **do**
for every marked bundle B_i out of P_j in G **do**
Let $S = \{g(l) | e_l \in B_i$ (note that each edge e_j is in a marked bundle)$\}$;
Schedule in X at time $t(i)$ the multicasting of the message associated with bundle B_i from processor P_j to the set of processors S (if $|S| = 1$, the operation is unicasting);
for every edge $e_l \in B_i$ in a marked bundle **do**
Add to the hold set of processor $g(l)$ (i.e., $H(g(l))$, in $\hat{G}$ message B_i;
Add the edge from $P_{g(l)}$ to P_q in $\hat{G}$, where q is the processor where edge e_l ends in G;
endfor
endfor
endfor
Split every multi-edge bundle in $(\hat{I}, \hat{G})$ into single-edge bundles;
end of Procedure

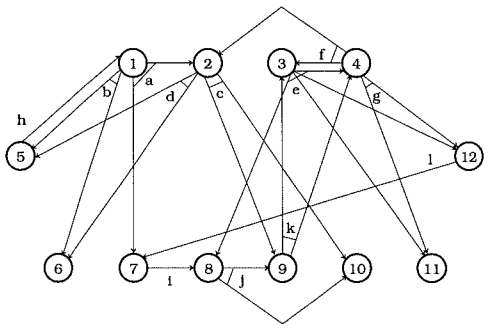

Figure 2: $l - MM_C$ problem instance (I, G).

We now apply our algorithm to the problem in Figure 2. The first loop transforms it to the problem in Figure 3. The bundles that are split are the ones emanating out of processors P_3, P_8, and P_9.

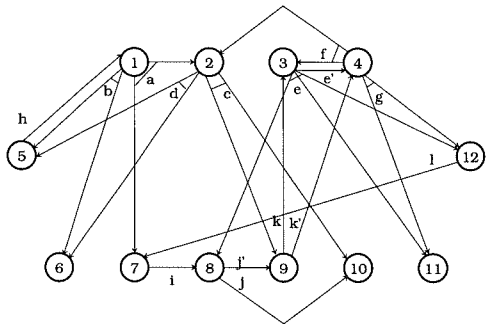

Figure 3: (I, G) after the first loop transformation.

In Figure 4 we show all the labelings performed by procedure l-FORWARD. The $\hat{F}_i$ and r_i values computed by the procedure are the numbers that appear on top of the vertices in Figure 4. The top number just below each of the vertices is the index of B_i, the next line shows the message name, and then the value $t(i)$. The line just above the bottom one has the e_j index, and the bottom number is the processor index to where that edge is to be forwarded.

The forwarding is: message a is multicasted to processors P_3 and P_5; message c is unicasted to processor P_6; message e is unicasted to processors P_7 and P_{10}; and message f is unicasted to processor P_{11}. The resulting problem, $(\hat{I}, \hat{G})$ is given in Figure 5. The difference now is that message a is to be transmitted form

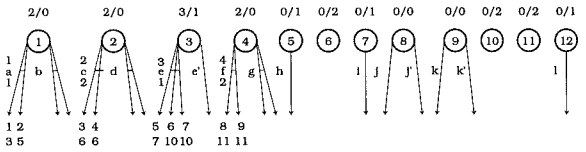

Figure 4: Labels generated by l-FORWARD.

processor P_3 to P_2, and from P_5 to P_7; message c is to be multicasted from P_6 to P_9 and P_{10}; message e is to be unicasted from P_7 to P_8 and multicasted from P_{10} to P_{11} and P_{12}; and message f is to be multicasted from P_{11} to P_2 and P_3. In the final transformation all the multi-edge bundles are replaced by single-edge bundles. In our examples the bundles for messages b, c, e, f, and g are replaced by two edge single-edge bundles.

We should point out that all the $t()$ values could be set to one in the above example and there would not be any conflicts and the total communication time needed by l-FORWARD would be decreased from two to one. In general this is not always possible. For example if processor P_{10} had one edge emanation out of it then processor P_3 would only forward one edge there and the other one would be forwarded to processor P_{11}. Processor P_4 would also need to forward an edge to P_{11}. Therefore, the two messages to be received by P_{11} would need to be sent at different times, so the above reduction in total communication time is not possible in this other problem instance.

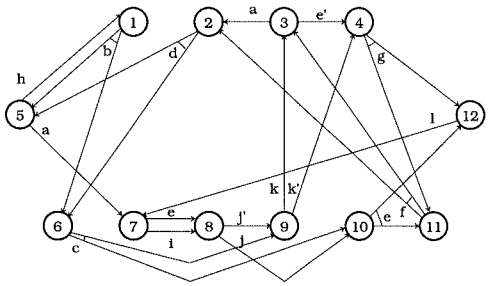

Figure 5: Resulting problem instance $(\hat{I}, \hat{G})$ just before last transformation (last line).

Theorem 2.1. Problem instance $(\hat{I}, \hat{G})$ is an instance of the MU_C problem and schedule X with total communication time $d - \lfloor \frac{d}{l} \rfloor + 1$ constructed by procedure l-FORWARD plus any communication schedule for $(\hat{I}, \hat{G})$ is a communication schedule for (I, G).

Proof. First we show that schedule X is a feasible schedule. From the definition of F_i and $\hat{F}_i$ we know that only those processors with more than d edges will be forwarding edges to other processors, and the tentative number of such edges is $G_i - d$. Since the edges to be forwarded are the ones from the bundles with the largest number of edges, it then follows that all the edges to be forwarded belong to at most $d - \lfloor \frac{d}{l} \rfloor$ of the bundles emanating out of the processor. From the definition of $t()$ we know that these messages will be multicasted at different times. Let α be the total number of edges emanating out of processor P_i and let k be the number of single-edge bundles emanating out of P_i. The total number of edges in multi-edge bundles is $\alpha - k$, and this value is greater than or equal to $\alpha - d$ (the tentative number of edges to be forwarded) since $k \leq d$. This together with the way we define $\hat{F}_i$ implies that all the edges to be forwarded belong to multi-edge bundles and all the edges in these multi-edge bundles will be forwarded. From the $g()$ labels and procedure l-FORWARD we know that each processor will receive at most d edges to be forwarded. Since the messages forwarded consist of at least two edges, except possibly for the one forwarded to the previous and to the next processor, it follows that at most $\lfloor \frac{d}{2} \rfloor + 1$ messages will be received by each processor. Since $\lfloor \frac{d}{2} \rfloor + 1 \leq d - \lfloor \frac{d}{l} \rfloor + 1$, these messages are labeled sequentially. Therefore, all of these messages arrive at different times and there are no conflicts. Note that because of this last discussion the total communication time is one unit more than what was expected.

From the function $g()$ and procedure l-FORWARD we note that each message is forwarded to the appropriate processor so that if we carry out all the communications given by the resulting problem instance $(\hat{I}, \hat{G})$, we also solve problem instance (I, G). Furthermore, the resulting problem is of degree d. Therefore, schedule X plus any communication schedule for $(\hat{I}, \hat{G})$ is a communication schedule for (I, G). []

Lemma 2.1: The time complexity for procedure l-FORWARD is $O(n + e)$, where e is the total number of edges in (I, G).

Proof. The steps before the nested for-loops and the last step take $O(n + e)$. Overall, the innermost loop is executed once for each edge in the bundle, the middle loop is executed once for each bundle emanating out of the processor, and the outermost loop is executed once for each processor. Therefore, the total time complexity is $O(n + e)$. []

A communication schedule with total communication time d for the instance of the multimessage unicasting problem $(\hat{I}, \hat{G})$ can be constructed in polynomial time by reducing the problem to the Makespan Openshop Preemptive Scheduling problem [8].

Lemma 2.2: ([6]) The above informal reduction can be used to construct communication schedule X' with total communication time equal to $\hat{d}$ for any multimessage unicasting problem $(\hat{I}, \hat{G})$ of degree $\hat{d}$ with $\hat{n}$ processors. The procedure takes $O(r(min\{r, \hat{n}^2\} + \hat{n}\ log\ \hat{n}))$ time, where r is the number of messages $(r \leq \hat{d}\hat{n})$.

Proof. The specifics of the reduction and the correct-

ness proof appears in [6]. []

Theorem 2.2: Communication schedule X generated by procedure l-FORWARD plus the communication schedule X' generated by the reduction in [6] is a communication schedule for (I, G) with total communication time $2d - \lfloor \frac{d}{l} \rfloor + 1$. The overall time complexity for our procedure is $O(r(min\{r, n^2\} + n\ log\ n))$, r is the number of messages ($r \leq dn$).

Proof. The proof follows from Lemmas 2.1, and 2.2, and Theorem 2.1. []

In order for the communication schedule to be executable by pr-dynamic networks one needs to perform all the multicasting operations to adjacent numbered processors. In Figure 4 we see that message a will be multicasted only processors P_3 and P_5, and message e to processors P_7 and P_{10}. Since no other processor will transmit at the same time to P_4, P_8, and P_9, then message a can be multicasted to processors P_3, P_4, and P_5, and message e to processors P_7, P_8, P_9, and P_{10}. By applying this type of transformation we can establish the following result.

Theorem 3.2: Communication schedule X plus X' is a communication schedule with total communication time $2d - \lfloor \frac{d}{l} \rfloor + 1$ for any pr-dynamic network.

Proof. The proof of this theorem follows form the fact that all multicasting messages can be sent to a set of adjacent processors and that such a multicasting operation can be performed in one step in pr-dynamic networks. []

3 Conclusion

When $1 < l < 2$ one can use a similar strategy to construct a communication schedule with total communication time $\lfloor (1 + \frac{l-1}{2})d \rfloor + 1$, for the $l - MMF_C$ problem. For brevity we do not include this result.

Our algorithm is designed for the off-line case when all the information is available in advance, which is justified by the following application. The author has also developed algorithms for the on-line case. Multimessage multicasting problems arise when solving sparse systems of linear equations via iterative methods (e.g., a Jacobi-like procedure), and most dynamic programming procedures in a parallel computing environment. Let us now discuss the application involving linear equations in more detail. We are given the vector $X(0)$ and we need to evaluate $X(t)$ for $t = 1, 2, \ldots$, using the iteration $x_i(t+1) = f_i(X(t))$. But since the system is sparse every f_i depends on very few terms. A placement procedure assigns each x_i to a processor where it will be computed at each iteration by evaluating $f_i()$. Good placement procedures assign a large number of $f_i()$s to the processor where the vector components it requires are being computed, and therefore can be computed locally. However, the remaining $f_i()$s need vector components computed by other processors. So at each iteration these components have to be multicasted (transmitted) to the set of processors that need them. The strategy is to compute $X(1)$ and perform the multimessage multicasting, then compute $X(2)$ and perform the multicasting, and so on. The same communication schedule is used at each iteration. Speedups of n for n processor systems may be achieved when the processing and communication load is balanced, by overlapping the computation and communication time.

References

[1] G. S. Almasi, and A. Gottlieb, *Highly Parallel Computing*, The Benjamin/Cummings Publishing, New York, 1994.

[2] E. G. Coffman, Jr, M. R. Garey, D. S. Johnson, and A. S. LaPaugh, "Scheduling File Transfers in Distributed Networks," *SICOMP*, 14(3) (1985), pp. 744 – 780.

[3] H.-A. Choi, and S. L. Hakimi, "Data Transfers in Networks," *Algorithmica*, Vol. 3, (1988), pp. 223 – 245.

[4] T. F. Gonzalez, "Multi-Message Multicasting," Proceedings of Irregular'96, Lecture Notes in CS (1117), Springer, (1996), pp. 217-228.

[5] T. F. Gonzalez, "Improved Multimessage Multicasting Approximation Algorithms," Proceedings of the Ninth PDCS'96, (1996), pp. 456 – 461, July 1996.

[6] T. F. Gonzalez, "Complexity and Approximations for MultiMessage Multicasting," *Journal of Parallel and Distributed Computing*, 55, (1998), 215 – 235.

[7] T. F. Gonzalez, "Simple Multimessage Multicasting Approximation Algorithms Allowing Forwarding," Proceedings of the Tenth PDCS'97, (1997), 372 – 377.

[8] T. F. Gonzalez, and S. Sahni, "Open Shop Scheduling to Minimize Finish Time," *JACM*, Vol. 23, No. 4, (1976), pp. 665 – 679.

[9] I. S. Gopal, G. Bongiovanni, M. A. Bonuccelli, D. T. Tang, and C. K. Wong, "An Optimal Switching Algorithm for Multibean Satellite Systems with Variable Bandwidth Beams," *IEEE Transactions on Communications*, COM-30, 11 (1982) pp. 2475 – 2481.

[10] B. Hajek, and G. Sasaki, "Link Scheduling in Polynomial Time," *IEEE Transactions on Information Theory*, Vol. 34, No. 5, (1988), pp. 910 – 917.

[11] S. C. Liew, "A General Packet Replication Scheme for Multicasting in Interconnection Networks," *Proceedings IEEE INFOCOM '95*, Vol.1 (1995), pp. 394 – 401.

[12] P. I. Rivera-Vega, R, Varadarajan, and S. B. Navathe, "Scheduling File Transfers in Fully Connected Networks," *Networks*, Vol. 22, (1992), pp. 563 – 588.

[13] H. Shen, "Efficient Multiple Multicasting in Hypercubes," *Journal of Systems Architecture*, Vol. 43, No. 9, Aug. 1997.

[14] J. S. Turner, "A Practical Version of Lee's Multicast Switch Architecture," *IEEE Transactions on Communications*, Vol. 41, No 8, (1993), pp. 1166 – 1169.

[15] J. Whitehead, "The Complexity of File Transfer Scheduling with Forwarding," *SICOMP* Vol. 19, No 2, (1990), pp. 222 – 245.