Nordic Journal of Computing 5(1998), 196–213.

BOUNDED FAN-OUT MULTIMESSAGE MULTICASTING

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

Abstract. We consider Multimessage Multicasting over the n processor complete (or fully connected) static network (MM_C). We present a fast approximation algorithm with an improved approximation bound for problem instances with small fan-out (maximum number of processors receiving any given message), but arbitrary degree d, where d is the maximum number of messages that each processor may send or receive. These problem instances are the ones that arise in practice, since the fan-out restriction is imposed by the applications and the number of processors available in commercial systems.

Our algorithm is centralized and requires all the communication information ahead of time. Applications where this information is available include iterative algorithms for solving linear equations and most dynamic programming procedures. The Meiko CS-2 machine as well as other computer systems whose processors communicate via dynamic networks will also benefit from our results at the expense of doubling the number of communication phases.

CR Classification: F.2.2, C.1.4, G.2.2, C.2.1, G.1.3

Key words: approximation algorithms, multimessage multicasting, dynamic networks, parallel iterative methods, generalized edge coloring

1. Introduction

1.1 The MM_C problem

The Multimessage Multicasting problem over the n processor complete static network (there are bidirectional links between every pair of processors), MM_C, consists of finding a communication schedule with least total communication time for multicasting (transmitting) any given set of messages. Specifically, there are n processors, $P = \{P_1, P_2, \ldots, P_n\}$, interconnected via a network N. Each processor is executing processes, and these processes are exchanging messages that must be routed through the links of N. 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. Our introduction is a condensed version of the one given by Gonzalez [1999], which includes a complete justification for the MM_C problem as well as motivations, applications and examples.

Received June 18, 1997; revised September 15, 1998; accepted October 2, 1998.

Routing in the complete static network (or simply a complete network) is the simplest and most flexible when compared to other static and dynamic networks. Multimessage Multicasting for dynamic networks (or multistage interconnection networks) that can realize all permutations and replicate data (e.g., n by n Benes network based on 2 by 2 switches that can also act as data replicators) is not too different, in the sense that the number of communication phases for these dynamic networks can be shown to be twice of that in the complete network. This is accomplished by translating each communication phase for the complete network into two communication phases for these dynamic networks. The first phase replicates data and transmits it to other processors, and the second phase distributes data to the appropriate processors [see Lee 1988, Liew 1995, Turner 1993]. The messages transmitted in each of these communication phases have to go through $O(\log n)$ switches on Benes networks based on 2 by 2 switches. The IBM GF11 machine [see Almasi and Gottlieb 1994], 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% [see Lee 1988, Liew 1995, Turner 1993]. In what follows we concentrate on the MM_C problem because it has a simple structure, and, as we mentioned before, results for this network can be easily translated to other dynamic networks.

Let us formally define our problem. Processor P_i needs to multicast m_i messages, each requiring one time unit to reach all of its destinations. The j^{th} message of processor P_i has to be sent to the set of processors $T_{i,j} \subseteq P - \{P_i\}$. Let r_i be the number of distinct messages that processor P_i should receive. We define the *degree* of a problem instance as $d = \mathbf{max}\{m_i, r_i\}$, i.e., the maximum number of messages that any processor sends or receives. We define the *fan-out* of a problem instance as $k = \mathbf{max}\{ \mid T_{i,j} \mid \}$, i.e., the maximum number of different processors that must receive any given message. Consider the following example.

EXAMPLE 1. There are three processors ($n = 3$). Processors P_1, P_2, and P_3 must transmit 3, 4 and 2 messages, respectively (i.e., $m_1 = 3, m_2 = 4$, and $m_3 = 2$). The destinations of these messages are given in Table I. For this example $r_1 = 4$, $r_2 = 4$, $r_3 = 4$, $d = 4$, and the fan-out is 2.

It is convenient to represent 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 $|T_{i,j}|$ directed edges or branches associated with each message are *bundled* together. The problem instance given in Example 1 is depicted in Fig. 1.1 as a directed multigraph.

TABLE I: Destination of messages for Example 1.

$T_{i,j}$	$j=1$	2	3	4
$i=1$	$\{2\}$	$\{3\}$	$\{2,3\}$	$\emptyset$
2	$\{1\}$	$\{1\}$	$\{3\}$	$\{1,3\}$
3	$\{1,2\}$	$\{2\}$	$\emptyset$	$\emptyset$

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

(1) During each time unit each processor may multicast a message to a set of processors; and

(2) During each time unit each processor may receive at most one message.

Our communication model allows us to transmit any of the messages in one or more stages. I.e., each set $T_{i,j}$ (or bundle) can be partitioned into subsets, and each of these subsets is transmitted at a different time. This added routing flexibility reduces the total communication time.

A *communication mode* C is a collection of subsets of branches from a subset of the bundles that obey the following communication rules imposed by our network:

(1) Branches may emanate from at most one of the bundles in each processor; and

(2) All of the branches end at different processors.

A *communication schedule* S for a problem instance I is a sequence of communication modes such that each branch in each message is in exactly one of the communication modes. The *total communication time* is the number of communication modes in the schedule, i.e., the latest time at which there is a communication. Our problem consists of constructing a communication schedule with least total communication time. From the communication rules we know that a 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. To simplify the analysis of our approximation algorithm we use this lower bound as the

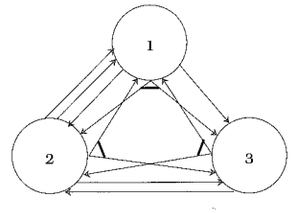

Fig. 1.1: Directed multigraph representation for Example 1. The thick line joins all the edges (branches) in the same bundle.

objective function value of an optimal solution. A communication schedule with total communication time equal to four for the problem instance given in Example 1 is given in Table II

TABLE II: Communication schedule for Example 1.

Time 1	$T_{1,1} : P_1 \to P_2$	$T_{2,4} : P_2 \to (P_1, P_3)$	-
Time 2	$T_{1,2} : P_1 \to P_3$	$T_{2,1} : P_2 \to P_1$	$T_{3,2} : P_3 \to P_2$
Time 3	$T_{1,3} : P_1 \to P_3$	$T_{2,2} : P_2 \to P_1$	$T_{3,1} : P_3 \to P_2$
Time 4	$T_{1,3} : P_1 \to P_2$	$T_{2,3} : P_2 \to P_3$	$T_{3,1} : P_3 \to P_1$

Using our multigraph representation one can visualize the MM_C problem as a generalized edge coloring directed multigraph (GECG) problem. This problem consists of coloring the edges with the least number of colors (positive integers) so that the communication rules (now restated in the appropriate format) imposed by our network are satisfied: (1) every pair of edges from different bundles emanating from the same vertex must be colored differently; and (2) all incoming edges to each vertex must be colored differently. The colors correspond to different time periods. In what follows we corrupt our notation by using interchangeably colors and time periods; vertices and processors; and bundles, branches or edges, and messages.

1.2 Previous work, new results, and applications

Gonzalez [1999] 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, i.e. the upper bound is also a lower bound. One observes that the lower bound applies when the fan-out is huge, and thus the number of processors is also huge. Since this environment is not likely to arise in the near future, we turn our attention in subsequent sections to important subproblems likely to arise in practice.

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. There will be no conflicts because $d = 1$, i.e., each processor may send at most one message and receive at most one message. When the processors are connected via a dynamic network whose basic switches allow data replication, the basic multicasting problem can be solved in two stages: the data replication step followed by the data distribution step [see Lee 1988, Turner 1993, Liew 1995]. This two stage process can be used in the Meiko CS-2 machine.

Gonzalez [1999] considered the case when each message has fixed fan-out k, and showed that when $k = 1$ the problem corresponds to the Makespan Openshop Preemptive Scheduling problem which can be solved in polynomial time [see Gonzalez and Sahni 1976]. In this case every degree d prob-

lem instance has a 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 one 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. Coffman *et al.* [1985] studied a version of the multimessage unicasting problem when messages have different lengths, each processor has $\gamma(P_i)$ ports each of which can be used to send or receive messages, and messages are transmitted without interruption (non-preemptive mode). Whitehead [1990] considered the case when messages can be sent indirectly. The preemptive version of these problems as well as other generalizations were studied by Choi and Hakimi [1987, 1988], Hajek and Sasaki [1988], and Gopal *et al.* [1982]. Some of these papers considered the case when the ports are not interchangeable, i.e., it is either an output port or an input port. Rivera-Vega *et al.* [1992] studied the file transferring problem, a version of the multimessage unicasting problem for the complete network when every vertex can send (receive) as many messages as the number of outgoing (incoming) links. The distributed version of the multimessage unicasting problem with forwarding, $DMUF_C$, corresponds to the h-relation problem in the area of optical-communication parallel computers [see Goldberg *et al.* 1997, Valiant 1990]. This distributed work is more general than ours, but the expected total communication time for this problem is $O(d + \log\log n)$ [see Goldberg *et al.* 1997]. With the exception of the work by Gonzalez [1999, 1996, 1996, 1997, 1998], and Shen [1997], research has been limited to unicasting and all known results about multicasting are limited to single messages. Shen [1997] has studied multimessage multicasting for hypercube connected processors. His procedures are heuristic and try to minimize the maximum number of hops, amount of traffic, and degree of message multiplexing. 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 CS2 machine, after solving basic synchronization problems with barriers.

The MM_C problem is significantly harder than the MU_C. Gonzalez [1999] showed that even when $k = 2$ the decision version of the MM_C problem is NP-complete. He also developed an algorithm to construct a communication schedule with total communication time $2d - 1$ for the case when the fan-out is two, i.e., $k = 2$. Gonzalez [1999] developed an $O(q \cdot d \cdot e)$ time algorithm, where $e \leq nd$ (the number of branches), to construct for problem instances of degree d a communication schedule with total communication time $qd + k^{\frac{1}{q}}(d-1)$, where q is the maximum number of colors that can be used to color each bundle and $2 \leq q < k$. Gonzalez [1997, 1998] has studied

the MM_C when messages can be sent indirectly (forwarding is allowed) and when all the message destinations are not known globally (distributed version). The algorithm in this paper is faster. The distributed work is more general, but the expected total communication time of those solutions is $O(d + \log n)$ [see Gonzalez 1998].

In this paper we present a fast approximation algorithm with an improved approximation bound for problem instances with any arbitrary degree d, but small fan-out. These problem instances are the ones that arise in practice, since the fan-out restriction is imposed by the applications and the number of processors available in commercial systems. We should point out that Section 2 does not depend on any results in previous papers. All of our approximation algorithms generate a coloration that uses at most $a_1 \cdot d + a_2$ colors. The value of a_1 for the different methods we have developed and for different values for k is given in Table III. The number inside the parenthesis that follows the method's names indicates the maximum number of different colors one may use to color the branches in each bundle. The methods labeled "Simple" are for the method developed by Gonzalez [1999]. The method labeled "Involved (2)" is the fastest and it is the one discussed in Section 2. The method labeled "With Matching" is similar to the one in this paper but uses matching [see Gonzalez 1996], and the one labeled "Best Bound" is the best one so far for small values of k. The proofs for these bounds are tedious because the equations and algorithms are more complex, but the proof technique is similar to the one for the method given in Section 2. All of these bounds are asymptotic to the ones reported in this paper. By selecting three colors per bundle instead of two colors, one can also improve the asymptotic bound reported in this paper (see Table III the line labeled "Improved (3)"). The algorithm is slower, its correctness proof is quite involved, but the proof for the approximation bound is similar in nature to the one in Section 2. The benefit when k is 15 is about 10% improvement and about 40% when k is 100. At the end of Section 2 we explain how we generate the entry labeled "Involved (2)" from our lemmas and theorem.

TABLE III: Value of a_1 for different methods.

Method \ k	3	4	5	7	10	15	20	50	100
Simple (2)	3.73	4.00	4.23	4.65	5.16	5.87	6.47	9.07	12.00
Involved (2)	3.33	3.50	3.60	4.43	4.60	5.53	6.00	8.56	11.54
With Matching (2)	2.67	3.00	3.50	4.29	4.50	5.47	6.00	8.54	11.53
Better Bound (2)	2.50	3.00	3.50	4.14	4.40	5.40	5.75	8.52	11.52
Simple (3)	–	–	4.00	4.55	4.81	5.27	5.60	6.67	7.62
Involved (3)	–	3.56	4.00	4.26	4.67	5.00	5.20	6.23	7.24
Simple (4)	–	–	5.50	5.63	5.78	5.97	6.11	6.66	7.16
Simple (5)	–	–	–	6.48	6.58	6.72	6.82	7.19	7.51

Multimessage multicasting problems arise when solving sparse systems of linear equations via iterative methods (e.g., a Jacobi-like procedure), most dynamic programming procedures, etc. Let us now discuss the application involving linear equations. 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 required multimessage multicasting, then compute $X(2)$ and perform the multicasting, and so on. The same communication schedule is used at each iteration, and can be computed off-line once the placement of the x_is has been decided. The same communication schedule can also be used to solve other systems with the same structure, but different coefficients. 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. This may be achieved by executing two concurrent tasks in each processor. One computes the x_is, beginning with the ones that need to be multicasted, and the other deals with the multicasting of the x_i values. If all the transmissions can be carried out by the time the computation of all the x_is is finished then we have achieved maximum performance. But if the communication takes too long compared to the computation, then one must try another placement or try alternate methods. Clearly, our solutions are machine dependent, but the solutions we generate have total communication time smaller than those for more general problems [see Gonzalez 1998].

2. Approximation algorithm

The input to our algorithm is a directed multigraph G with bundled edges, integers h and l that restrict the color selection process and it is assumed that $(k > l > h \geq 1)$. The different values for l and h generate solutions with a different number of colors. There is no simple formula to determine a optimal choice of l and h; however, $l = k-1$ and $h \approx \lfloor \sqrt{k-1} \rfloor$ generate near optimal solutions. Note that k and d can be easily extracted from the multigraph. The algorithm colors the edges emanating out of P_1, then P_2, and so on until P_n. Each bundle is colored with at most two different colors. When coloring the branches emanating out of P_j, each of these branches leads to a processor with at most $d-1$ other edges incident to it, some of which have already been colored. These colors are called t_{j-1}-*forbidden* with respect to a given branch emanating from P_j. When considering processor P_j, a

t_{j-1}-forbidden color with a special property is selected from each bundle and then such color is used to color as many of the branches of the bundle. The remaining uncolored branches are colored with a second color whose existence is guaranteed by setting the total number of colors available to an appropriate value. Before we present our algorithm we define some useful terms.

At the beginning of the j^{th} iteration the algorithm has colored all the branches emanating from processors $P_1, P_2, \ldots, P_{j-1}$. Let us define the following terms from this partial coloration. For $0 \leq i \leq k$, let C_i^b be the set of colors that are t_{j-1}-forbidden in exactly i branches of bundle b. Let $c_i^b = |C_i^b|$. When the set b is understood, we will use c_i for c_i^b, and C_i for C_i^b. Since there can be at most $d-1$ t_{j-1}-forbidden colors in each branch and there are at most k branches in each bundle, it then follows that $\sum_{i=1}^{k} ic_i^b \leq (d-1)k$ for each bundle b emanating from P_j. Clearly, all the branches of bundle b can be colored with any of the colors in C_0^b that have not been used to color other branches emanating from P_j. Also, one can color all the branches of bundle b with two colors, $p \in C_i^b$ and $q \in C_{i'}^b$, for $i \neq i'$, provided that colors p and q are not t_{j-1}-forbidden in the same branch of bundle b, and have not been used to color another branch emanating from processor P_j. We say that a color is s_j-*free* if such color has not yet been used to color any of the branches emanating from processor P_j. Note that a color stops being s_j-free as soon as it is used to color a branch emanating out of P_j.

To simplify our notation we define the expressions L and R as follows

$$L = \frac{h^2+h+2}{2} + \frac{l}{d-1} - \frac{h^2+h-2}{2(d-1)}, \text{ and } R = (h+1)^2 + \frac{(h+1)(h^2+3h)}{2(l-h)} + \frac{-2lh^2+h^3+h}{2(d-1)(l-h)}$$

Procedure Coloring is defined for all $d \geq \frac{2l+2h^2}{h(h+3)}$, $k \geq L$, $k > l > h \geq 1$ and $d \geq 4$. These preconditions might give the feeling that there is a large number of cases for which our algorithm is not defined, but this is not the case because for each $k \geq 3$ there is a nonempty set of h and l values for which it is defined. We begin by establishing in Lemma 1 that $L \leq R$. This fact will be used to partition in two cases the set of values for which our algorithm is defined.

LEMMA 1. *For the set of values Procedure Coloring is defined* $L \leq R$.

PROOF. The proof of this lemma is in the Appendix, since it involves simple algebraic manipulations of long expressions. □

In Table IV we define equations eq.(0), ..., eq.($h+1$) that are used by the algorithm and are necessary for the correctness proof (Theorem 1). Let us now briefly outline our Procedure Coloring as well as some of the arguments used in the correctness proof. When coloring the bundles emanating out of processor P_j, Procedure Coloring finds the smallest integer q_b such that equation eq.(q_b) holds. Lemma 5 shows that at least one of such equation holds for each bundle b. Then r_b is defined as $min\{q_b, h\}$. For each bundle b

TABLE IV: Equations eq.(0), ..., eq.$(h+1)$.

	$c_0 \geq d;$	$eq.(0)$
for $1 \leq t \leq h$	$\sum_{i=0}^{t} c_i \geq (t+2)d - 2t;$ *or*	$eq.(t)$
	$\sum_{i=0}^{l} c_i \geq (h+2)d - 2h.$	$eq.(h+1)$

emanating from processor P_j an s-free color from $C_0^b, C_1^b, \ldots, C_{r_b}^b$ is selected to color as many branches in bundle b as possible. Lemma 4 can be used to show that one such color exists. The integer w_b is defined as q_b if $0 \leq q_b \leq h$ and it is set to l otherwise. The remaining uncolored branches of each bundle b are colored with an s-free color in $C_0^b, C_1^b, \ldots, C_{w_b}^b$. One can show the existence of one such color from Lemma 5.

Procedure Coloring is given in the next page. For the set of valid inputs defined above, the procedure computes the maximum number of colors needed (Δ) and a coloration for G with at most Δ colors.

To establish that Procedure Coloring generates a valid coloration for the cases it is defined is difficult. Theorem 1 establishes that our algorithm generates valid colorations for all valid inputs, and that it takes $O(ed)$, where e is the total number of edges. The proof of this theorem is based on Lemmas 4 and 5. Lemma 5 established that at least one of the equations $eq.(t)$ holds for each bundle emanating out of processor P_j, and Lemma 4 is used to show that one color from each bundle can be selected to color a subset of its branches. These lemmas are then used to show that a second color exists to color the remaining branches of each partially colored bundle. Lemma 3 is used in the proof of Lemmas 4 and 5, and requires Lemma 2.

The purely mechanical proofs (or parts of proofs) appear in the Appendix. Mathematica programs (and their outputs) for all of these mechanical proofs were developed by Gonzalez [1996].

LEMMA 2. *For the set of values Procedure Coloring is defined*

$$R \geq (h+1)^2 - \frac{h^2}{d-1} + \frac{h+1}{d-1}.$$

PROOF. The proof of this lemma is in the Appendix, since it involves simple algebraic manipulations of long expressions. □

Procedure Coloring (G, h, l)
/* Note that k, d, L, and R can be easily computed from G, h and l. */
/* Assume that $d \geq \frac{2l+2h^2}{h(h+3)}$, $k \geq L$, $k > l > h \geq 1$, and $d \geq 4$ */
case
:$R \leq k$: $\Delta = \frac{d(k+h+1)-(k+h)}{h+1}$;
:$L \leq k < R$: $\Delta = \frac{((2d-4)h+4d-2)l+2(d-1)k+(2-d)h^2+(d-2)h+2d}{2(l+1)}$;
endcase
/* The correctness of our procedure will be established in Theorem 1.*/
for $j = 1$ **to** n **do** /* For each processor P_j. */
for $b = 1$ **to** m_j **do** /* For each bundle emanating out of P_j.*/
compute $C_0^b, C_1^b, C_2^b, \ldots, C_k^b$;
let q_b be the smallest integer such that equation eq.(q_b) holds;
let $r_b =$ **min** $\{q_b, h\}$;
let $w_b = q_b$ if $0 \leq q_b \leq h$ and $w_b = l$ otherwise;
endfor
/* Color a subset of edges emanating from each bundle of P_j */
for each uncolored bundle b of P_j **do**; /* **for-loop-a** */
color as many branches of bundle b with one s_j-free color
in $C_0^b, C_1^b, \ldots, C_{r_b}^b$;
/* The above statement and the definition of the C sets implies
that the color comes out of the smallest indexed set with
an s_j-free color;*/
/* Color the remaining uncolored edges emanating from P_j */
for each partially colored bundle b of P_j **do**; /* **for-loop-b** */
color all uncolored branches of b with an s_j-free color in
$C_0^b, C_1^b, \ldots, C_{w_b}^b$;
endfor;
end of Procedure Coloring

LEMMA 3. *The value for Δ defined by Procedure Coloring is greater than or equal to $(h+2)d - 2h$.*

PROOF. The proof of this lemma is in the Appendix, since it involves simple algebraic manipulations of long expressions. □

LEMMA 4. *At the beginning of the j^{th} iteration of Procedure Coloring each bundle b emanating from processor P_j satisfies $\sum_{i=0}^{h} c_i^b \geq d$.*

PROOF. Suppose not. Suppose that there is a bundle b for which $\sum_{i=0}^{h} c_i^b < d$. Let bundle b emanating from processor P_j be a counter-example to the lemma with the least value for $\sum_{i=0}^{h} c_i^b$. In case of ties select the bundle b emanating from P_j with the largest value for c_0^b. If ties persist, then select any bundle with the two properties given above. In what follows we drop the superscript b, since b is fixed.

If $\sum_{i=1}^{h} c_i > 1$, one can easily find another counter-example with smaller $\sum_{i=0}^{h} c_i$, or one with the same $\sum_{i=0}^{h} c_i$ but larger value for c_0. In either case we contradict the properties of our counter-example. So we can assume without loss of generality that $\sum_{i=1}^{h} c_i \leq 1$.

Let $x = \sum_{i=0}^{h} c_i \leq d-1$. If $x = 0$ then t is set to zero; otherwise, t is set to the largest value in $[0, h]$ such that $c_t \neq 0$. From Lemma 3 we know that $\Delta \geq (h+2)d - 2h$, and since $h \geq 1$ and $d > 1$, we know that $\Delta \geq (h+2)d - 2h > d$. Since the remaining t_{j-1}-forbidden colors must be in $C_{h+1}, C_{h+2} \ldots, C_k$, it then follows that

$$\sum_{i=0}^{k} i \cdot c_i \geq t + (\Delta - x)(h+1).$$

Substituting $x = \sum_{i=0}^{h} c_i \leq d-1$ in the above inequality, we know that

$$\sum_{i=0}^{k} i \cdot c_i \geq t + (\Delta - (d-1))(h+1).$$

To complete the proof of the Lemma we need to show that the above expression is greater than $k(d-1)$, which contradicts that fact that $\sum_{i=0}^{k} i \cdot c_i \leq k(d-1)$. Since the remaining part of the proof is purely mechanical it appears in the Appendix.

LEMMA 5. *At the beginning of the j^{th} iteration of Procedure Coloring each bundle b emanating from processor P_j satisfies at least one of the inequalities eq.(t), for $0 \leq t \leq h+1$, holds.*

PROOF. Suppose not. Suppose that there is a bundle b for which all the above inequalities do not hold, i.e., $c_0^b < d$; $\sum_{i=0}^{t} c_i^b < (t+2)d - 2t$ for all $1 \leq t \leq h$; and $\sum_{i=0}^{l} c_i^b < (h+2)d - 2h$. Let bundle b emanating from processor P_j be a counter-example to the lemma with the least value for $\sum_{i=0}^{l} c_i$. In case of ties select a bundle b emanating from P_j with the least value for $\sum_{i=0}^{l} c_i$ that is largest in lexicographic order. I.e., c_0^b has the largest value; in case of ties, c_1^b is the largest value; in case of ties, c_2^b is the largest value; in case of ties, and so on. Since b is fixed, in what follows we drop the superscript b.

We define the vector $(d_0, d_1, \ldots, d_k)$ as follows:

$$d_0 = d-1, d_1 = 2d-2, d_2 = d-2, d_3 = d-2, d_4 = d-2, \ldots,$$
$$d_h = d-2, d_{h+1} = 0, d_{h+1} = 0, d_{h+2} = 0, \ldots,$$
$$d_l = 0, d_{l+1} = \Delta - ((h+2)d - 2h - 1), d_{l+2} = 0, \ldots, \text{ and } d_k = 0.$$

Note that by Lemma 3, we know that $d_{l+1} \geq 1$. The subvector $(c_1, c_2, \ldots, c_h)$ is either the subvector $(d_1, d_2, \ldots, d_h)$, or it is such that $c_i = d_i$ for all $0 \leq i \leq f < h$, $c_{f+1} < d_{f+1}$ and $c_{f+2}, \ldots, c_h$ are zero except possibly for one c_t such that $f+2 \leq t \leq h$. Note that if this is not the case then either the problem instance satisfies one of the inequalities, or it does not satisfy any of the inequalities but either it is not one with least $\sum_{i=0}^{l} c_i$ or it is one

with least $\sum_{i=0}^{l} c_i$, but not the largest in lexicographic order. In all cases we contradict the properties of the bundle b selected.

By definition, $\sum_{i=0}^{k} i \cdot c_i \leq k(d-1)$. We now establish a contradiction by showing that $\sum_{i=0}^{k} i \cdot c_i > k(d-1)$. It is simple to see that $\sum_{i=0}^{k} i \cdot c_i \geq \sum_{i=0}^{k} i \cdot d_i$. From the definition of the D vector, and $\Delta \geq (h+2)d - 2h$ (Lemma 3), we know that

$$\sum_{i=0}^{k} i \cdot d_i = 2d - 2 + \frac{(d-2)(h^2+h-2)}{2} + (\Delta - ((h+2)d - 2h - 1))(l+1).$$

To complete the proof of the Lemma we need to show that the above expression is greater than $k(d-1)$, which contradicts that fact that $\sum_{i=0}^{k} i \cdot c_i \leq k(d-1)$. Since the remaining part of the proof is purely mechanical it appears in the Appendix. □

THEOREM 1. *Procedure Coloring generates a communication schedule with total communication time at most the value of Δ computed by the algorithm, for every instance for which the Procedure is defined. The time complexity of the procedure is $O(ed)$, where e is the total number of edges.*

PROOF. First we prove that Procedure Coloring colors all the edges in the multigraph with Δ colors, where Δ is determined by the algorithm. Then we establish the time complexity bound.

Consider now the iteration for P_j for any $1 \leq j \leq n$. By Lemma 5 we know that at least one of the equations eq.(t) for $0 \leq t \leq h+1$ holds for each bundle emanating from P_j. Therefore, all the q_b values are integers in the range $[0,h+1]$, and all the r_b values are integers in the range $[0,h]$.

We now claim that one can color a nonempty subset of branches from each bundle with a distinct s-free color in $C_0^b, C_1^b, \ldots, C_{q_j}^b$. We prove this by showing that $\sum_{i=0}^{r_b} c_i^b \geq d$, since this fact guarantees that one unique s-free color in $C_0^b, C_1^b, \ldots, C_{q_j}^b$ for each bundle b can be selected in for-loop-a to color a nonempty subset of edges emanating out of each bundle. As we established before, $r_b \leq h$. If $r_b = h$ then by Lemma 4 it follows that $\sum_{i=0}^{r_b} c_i^b \geq d$. On the other hand, if $r_b < h$ then by definition of r_b and Lemma 5 we know that $eq.(r_b)$ holds. This implies that either $c_0 \geq d$ or $\sum_{i=0}^{r_b} c_i \geq (r_b+2)d - 2r_b$. Since $d > 2$, it then follows that $\sum_{i=0}^{r_b} c_i^b \geq d$. Therefore, in for-loop-a one can select unique s-free color in $C_0^b, C_1^b, \ldots, C_{q_j}^b$ for each bundle b to color a nonempty subset of edges emanating out of each bundle.

We now claim that at each iteration in the for-loop-b one can select unique colors to color the remaining uncolored branches of each bundle. From the definition of w_b we know that $\sum_{i=0}^{w_b} c_i \geq (w_b+2)d - 2w_b$. The number of colors that were t_{j-1}-forbidden in the same branch as the color selected in for-loop-b is at most $(d-2) \cdot r_b$, and the maximum number of colors used during for-loop-a and for-loop-b is at most $2d-1$. It follows that the colors that one can use to color the remaining branches are at least

TABLE V: Value of a_1 for different methods.

k	a_1	l	h	L	R
3	3.33	2	1	2.11	7.95
4	3.50	3	1	2.16	5.95
5	3.60	4	1	2.21	5.28
7	4.43	6	2	4.21	12.50
10	4.60	9	2	4.37	10.91
15	5.53	14	3	7.47	18.74
20	6.00	11	3	7.32	19.95
50	8.56	49	6	23.53	51.37
100	11.54	99	9	48.89	101.53

$(w_b+2)d-2w_b-(d-2)\cdot r_b-2d+1$. This is equivalent to $(d-2)(w_b-r_b)+1$. Since $d > 2$ and $w_b \geq r_b$, we know that there is at least one color left with which we can color all the remaining uncolored branches. This completes the correctness proof.

It is simple to see that the time complexity of the algorithm is bounded by $O(ed)$, where e is the total number of edges in the graph. This follows from the observation that each edge is considered a constant number of times and each time the algorithm spends $O(d)$ time on it. □

For $d = 20$ and different values for k, Table V gives the value for h and l that minimizes a_1. Table V also gives the values of L and R. Note that for $d < 20$ some of the a_1 values are smaller than the ones in Table V.

3. Discussion

We have developed exact and approximation algorithms for problem instances with small d and small k, but for brevity we do not include these special results. We have also studied the multimessage multicasting problem with α buffers and the result are schedules whose total communication time is about $1/\alpha$ of the total communication time of the schedules without buffering. However, buffering complicates the hardware and it is not available on the Meiko CS-2 machine. The problem is equally difficult with or without buffering, provided that only a fixed number of buffers are available.

Appendix

LEMMA 1. For the set of values Procedure Coloring is defined $L \leq R$.

PROOF. By applying simple algebraic transformations one can readily see that the definition of L is equivalent to

$$L = (h+1)^2 - \tfrac{h^2+3h}{2} + \tfrac{l}{d-1} - \tfrac{h^2+h-2}{2(d-1)}, \text{ and}$$
$$L = (h+1)^2 + \tfrac{-(h^2+3h)d+2h+2+2l}{2(d-1)}.$$

Substituting $l \leq \frac{h(h+3)d}{2} - h^2$ in the above inequality, and simplifying we obtain

$$L \leq (h+1)^2 + \tfrac{-2h^2+2h+2}{2(d-1)}.$$

Since $l > h$ we multiply the last term of the inequality by $\frac{l-h}{l-h}$ and obtain

$$L \leq (h+1)^2 - \tfrac{2h^2l}{2(d-1)(l-h)} + \tfrac{h(h+1)(h+3)d-2h^2(h+1)}{2(d-1)(l-h)} + \tfrac{2h^3-2h^2-2h}{2(d-1)(l-h)}.$$

After simple algebraic manipulations we get

$$L \leq (h+1)^2 - \tfrac{2h^2l}{2(d-1)(l-h)} + \tfrac{(h+1)h(h+3)}{2(l-h)} + \tfrac{h^3+h}{2(d-1)(l-h)} = R.$$

This completes the proof of the Lemma. □

LEMMA 2. For the set of values Procedure Coloring is defined

$$R \geq (h+1)^2 - \frac{h^2}{d-1} + \frac{h+1}{d-1}.$$

PROOF. Since $l > h$ we multiply the last two terms in left-hand side (LHS) of the inequality by $\frac{2(l-h)}{2(l-h)}$ and obtain

$$LHS = (h+1)^2 - \tfrac{2lh^2-2h^3}{2(d-1)(l-h)} + \tfrac{(h+1)2(l-h)}{2(d-1)(l-h)}.$$

Substituting $l \leq \frac{h(h+3)d}{2} - h^2$ in the rightmost l of the above inequality, we know

$$LHS \leq (h+1)^2 - \tfrac{2lh^2-2h^3}{2(d-1)(l-h)} + \tfrac{(h+1)(h(h+3)d-2h^2-2h)}{2(d-1)(l-h)}.$$

This is equivalent to

$$LHS \leq (h+1)^2 + \tfrac{h(h+1)(h+3)}{2(l-h)} + \tfrac{-2lh^2+h^3+h}{2(d-1)(l-h)} = R.$$

This completes the proof of the Lemma. □

LEMMA 3. The value for Δ defined by Procedure Coloring is greater than or equal to $(h+2)d-2h$.

PROOF. By Lemma 1 we partition the proof into the following two cases depending on the relative values of k, L and R.

Case 1: $k \geq R$.

Substituting the inequality in Lemma 2 in the equation for the conditions of Case 1, we know that $k \geq (h+1)^2 - \frac{h^2}{d-1} + \frac{h+1}{d-1}$. The resulting expression after multiplying both sides by $d-1$ and rearranging terms is

$$k(d-1) \geq (h^2+2h+1)d - h(2h+1).$$

Adding $d(h+1)-h$ to both sides and rearranging terms,

$$k(d-1) + h(d-1) + d \geq (h^2+3h+2)d - 2h(h+1).$$

Factoring $h+1$ in the right hand side of the inequality, rearranging terms, and substituting for the value of Δ determined by the procedure, we know

$$\Delta = \frac{(k+h+1)d-(k+h)}{h+1} \geq (h+2)d-2h.$$

This completes the proof for Case 1.

Case 2: $L \leq k < R$.

From the conditions of Case 2, we know that

$$k \geq \frac{h^2+h+2}{2} - \frac{h^2+h-2}{2(d-1)} + \frac{l}{d-1}.$$

This is equivalent to

$$2k(d-1) \geq d(h^2+h+2) - (2h^2+2h) + 2l,$$

which is equivalent to

$$2k(d-1) - (h^2-h-2)d + (2h^2-2h) - 2l \geq 2(h+2)d - 4h.$$

Adding $2(h+2)dl - 4hl$ to both sides of the inequality, distributing terms, and replacing for the value of Δ determined by the procedure we know that

$$\Delta = \frac{((2d-4)h+4d-2)l+2(d-1)k+(2-d)h^2+(d-2)h+2d}{2(l+1)} \geq (h+2)d-2h.$$

This completes the proof for Case 2 and the Lemma.

PROOF. Remaining Part of Lemma 4.

$$\sum_{i=0}^{k} i \cdot c_i \geq t + (\Delta - (d-1))(h+1) > k(d-1).$$

By Lemma 1 we partition the proof into the following two cases depending on the relative values of k, L and R.

Case 1: $k \geq R$.

Substituting the value $\Delta = \frac{d(k+h+1)-(k+h)}{h+1}$ computed by the algorithm for Case 1 in the above inequality

$$\sum_{i=0}^{k} i \cdot c_i \geq t + (\tfrac{d(k+h+1)-(k+h)}{h+1} - (d-1))(h+1).$$

This reduces to

$$\sum_{i=0}^{k} i \cdot c_i \geq t + d(k+h+1) - (k+h) - (d-1)(h+1),$$

which is equivalent to

$$\sum_{i=0}^{k} i \cdot c_i \geq t + k(d-1) + 1.$$

Since $t \geq 0$, we know that $\sum_{i=0}^{k} i \cdot c_i > k(d-1)$. This contradicts the definition of the c_is, and completes the proof for Case 1.

Case 2: $L \leq k < R$.

Substituting the value $\Delta = \frac{((2d-4)h+4d-2)l+2(d-1)k+(2-d)h^2+(d-2)h+2d}{2(l+1)}$ computed by the algorithm for Case 2 in the above inequality

$$\sum_{i=0}^{k} ic_i \geq t + \tfrac{k(h+1)(d-1)}{l+1} + \tfrac{2dl(h^2+2h+1)-2l(2h^2+2h)-d(h^3-1)+2(h^3+1)}{2(l+1)}.$$

Adding and subtracting $\frac{2(l+1)}{2(l+1)} = 1$, the equation becomes

$$\sum_{i=0}^{k} ic_i \geq t + \tfrac{k(h+1)(d-1)}{l+1} + \tfrac{2dl(h^2+2h+1)-2l(2h^2+2h+1)-d(h^3-1)+2h^3}{2(l+1)} + 1.$$

Factoring from the third term $2(d-1)(l-h)$, the expression becomes

$$\sum_{i=0}^{k} ic_i \geq t + \tfrac{k(h+1)(d-1)}{l+1} + ((h+1)^2 + \tfrac{d(h^3+4h^2+3h)-2lh^2+2(-2h^2-h)}{2(d-1)(l-h)})\tfrac{(d-1)(l-h)}{l+1} + 1.$$

Factoring $(d-1)$ from the fourth term, the expression becomes

$$\sum_{i=0}^{k} ic_i \geq t + \tfrac{k(h+1)(d-1)}{l+1} + ((h+1)^2 + \tfrac{h(h+1)(h+3)}{2(l-h)} + \tfrac{-2lh^2+h^3+h}{2(d-1)(l-h)})\tfrac{(d-1)(l-h)}{l+1} + 1.$$

By the conditions of the case we know that

$$k < R = (h+1)^2 + \frac{h(h+1)(h+3)}{2(l-h)} + \frac{-2lh^2+h^3+h}{2(d-1)(l-h)}.$$

Substituting in the above inequality and simplifying,

$$\sum_{i=0}^{k} i \cdot c_i > t + \tfrac{k(h+1)(d-1)}{l+1} + \tfrac{k(d-1)(l-h)}{l+1} + 1,$$

which is equivalent to

$$\sum_{i=0}^{k} i \cdot c_i > t + k(d-1) + 1.$$

Since $t \geq 0$, we know that $\sum_{i=0}^{k} i \cdot c_i > k(d-1)$. This contradicts the definition of the c_is, and completes the proof for Case 2 and the Lemma.

PROOF. Remaining Part of Lemma 5.

$$\sum_{i=0}^{k} i \cdot d_i = 2d-2+\tfrac{(d-2)(h^2+h-2)}{2}+(\Delta-((h+2)d-2h-1))(l+1) > k(d-1).$$

By Lemma 1 we partition the proof into the following two cases depending on the relative values of k, L and R.

Case 1: $R \leq k$.

Substituting the value for Δ and reordering terms the expression becomes,

$$\sum_{i=0}^{k} i \cdot d_i = \frac{2k(d-1)(l+1)-2ld(h+1)^2+dh(h^2-1)+2l(2h^2+2h+1)-2h^3}{2(h+1)}.$$

Adding and subtracting 2(h+1) to the numerator

$$\sum_{i=0}^{k} i \cdot d_i = \frac{2k(d-1)(l+1)-2ld(h+1)^2}{2(h+1)} + \frac{dh(h^2-1)+2l(2h^2+2h+1)+2(-h^3+h+1)+2(h+1)}{2(h+1)}.$$

Multiplying the right hand side by $(\frac{(d-1)(l-h)}{h+1})(\frac{h+1}{(d-1)(l-h)}) = 1$ we know that

$$\sum_{i=0}^{k} i \cdot d_i = \frac{(d-1)(l-h)}{h+1} \cdot (\frac{k(l+1)}{(l-h)} + \frac{-2ld(h+1)^2+dh(h^2-1)+2l(2h^2+2h+1)-2h^3}{2(d-1)(l-h)} + \frac{h+1}{(d-1)(l-h)}).$$

Substituting the conditions for Case 1

$$(k \geq R = \frac{2ld(h+1)^2 - dh(h^2-1) - 2l(2h^2+2h+1) + 2h^3}{2(d-1)(l-h)})$$

in the above equation, we know that

$$\sum_{i=0}^{k} i \cdot d_i \geq \frac{(d-1)(l-h)}{h+1}(\frac{k(l+1)}{(l-h)} - k + \frac{h+1}{(d-1)(l-h)}),$$

which is equivalent to

$$\sum_{i=0}^{k} i \cdot d_i \geq \frac{(d-1)(l-h)}{h+1}(\frac{k(h+1)}{(l-h)} + \frac{h+1}{(d-1)(l-h)}),$$

and thus

$$\sum_{i=0}^{k} i \cdot d_i \geq k(d-1)+1.$$

Therefore, $\sum_{i=0}^{k} i \cdot c_i \geq \sum_{i=0}^{k} i \cdot d_i > k(d-1)$. A contradiction. This completes the proof for Case 1.

Case 2: $L \leq k < R$.

Substituting the value for Δ and reordering terms the expression becomes

$$\sum_{i=0}^{k} i \cdot d_i = 2d - 2 + \frac{(d-2)(h^2+h-2)}{2} + \frac{2(d-1)k+(2-d)h^2+(2-d)h-2d+2}{2},$$

which is equivalent to

$$\sum_{i=0}^{k} i \cdot d_i = k(d-1)+1.$$

Therefore, $\sum_{i=0}^{k} i \cdot c_i \geq \sum_{i=0}^{k} i \cdot d_i > k(d-1)$. A contradiction. This completes the proof for Case 2 and the Lemma.

References

ALMASI, G. S. AND GOTTLIEB, A. 1994. *Highly Parallel Computing*, 2nd edition. The Benjamin/Cummings Publishing Co., Inc., New York.

CHOÏ, H. A. AND HAKIMI, S. L. 1987. Data Transfers in Networks with Transceivers. *Networks 17*, 393–421.

CHOI, H. A. AND HAKIMI, S. L. 1988. Data Transfers in Networks. *Algorithmica 3*, 223–245.

COFFMAN, E. G., GAREY, M. R., JOHNSON, D. S., AND LAPAUGH, A. S. 1985. Scheduling File Transfers in Distributed Networks. *SIAM J. on Computing 14*, 3, 744–780.

GOLDBERG, L. A., JERRUM, M., LEIGHTON, T., AND RAO, S. 1997. Doubly Logarithmic Communication Algorithms for Optical-Communication Parallel Computers. *SIAM J. Computing 26*, 4, 1100–1119.

GONZALEZ, T. F. 1996. Improved Multimessage Multicasting Approximation Algorithms. In *Proceedings of the Ninth International Conference on Parallel and Distributed Computing Systems*, 456–461. Also, Technical Report TRCS-96-16, UCSB, Dept. of Computer Science.

GONZALEZ, T. F. 1996. Proofs for Improved Approximation Algorithms for Multimessage Multicasting. Technical Report TRCS-96-17, UCSB, Dept. of Computer Science.

GONZALEZ, T. F. 1997. Algorithms for Multimessage Multicasting With Forwarding. In *Proceedings of the Tenth International Conference on Parallel and Distributed Computing Systems*, 372–377. Also, Technical Report TRCS-97-24, UCSB, Dept. of Computer Science.

GONZALEZ, T. F. 1998. Distributed Algorithms for Multimessage Multicasting. Technical Report TRCS-98-23, UCSB, Dept. of Computer Science.

GONZALEZ, T. F. 1999. Complexity and Approximations for Multimessage Multicasting. *Journal of Parallel and Distributed Computing* (to appear). Also, Technical Report TRCS-96-15, UCSB, Dept. of Computer Science.

GONZALEZ, T. F. AND SAHNI, S. 1976. Open Shop Scheduling to Minimize Finish Time. *JACM 23*, 4, 665–679.

GOPAL, I. S., BONGIOVANNI, G., BONUCCELLI, M. A., TANG, D. T., AND WONG, C. K. 1982. An Optimal Switching Algorithm for Multibean Satellite Systems with Variable Bandwidth Beams. *IEEE Transactions on Communications 30*, 11, 2475–2481.

HAJEK, B. AND SASAKI, G. 1988. Link Scheduling in Polynomial Time. *IEEE Transactions on Information Theory 34*, 5 (Sept.), 910–917.

LEE, T. T. 1988. Non-blocking Copy Networks for Multicast Packet Switching. *IEEE J. Selected Areas of Communication 6*, 9 (Dec.), 1455–1467.

LIEW, S. C. 1995. A General Packet Replication Scheme for Multicasting in Interconnection Networks. In *Proceedings IEEE INFOCOM*, Volume 1, 394–401.

RIVERA-VEGA, P. I., VARADARAJAN, R., AND NAVATHE, S. B. 1992. Scheduling File Transfers in Fully Connected Networks. *Networks 22*, 563–588.

SHEN, H. 1997. Efficient Multiple Multicasting in Hypercubes. *Journal of Systems Architecture 43*, 9 (Aug.), 655–662.

TURNER, J. S. 1993. A Practical Version of Lee's Multicast Switch Architecture. *IEEE Transactions on Communications 41*, 8 (Aug.), 1166–1169.

VALIANT, L. G. 1990. General Purpose Parallel Architectures. In *Handbook of Theoretical Computer Science*. Elsevier, New York, 943–971.

WHITEHEAD, J. 1990. The Complexity of File Transfer Scheduling with Forwarding. *SIAM Journal on Computing 19*, 2 (Apr.), 222–245.