Multi-Message Multicasting

Teofilo F. Gonzalez

Department of Computer Science,
University of California, Santa Barbara, CA, 93106, USA

Abstract. We consider the Multi-Message Multicasting problem for the n processor fully connected static network. We present an efficient algorithm to construct a communication schedule with total communication time at most d^2, where d is the maximum number of messages a processor may send (receive). We present an algorithm to construct for any problem instance of degree d and fan-out k (maximum number of processors that may receive a given message) a communication schedule with total communication time at most $qd + k^{\frac{1}{q}}(d-1)$, for any integer $q \geq 2$. The time complexity bound for our algorithm is $O(n(d(q+k^{\frac{1}{q}}))^q)$. Our main result is a linear time approximation algorithm with a smaller approximation bound for small values of k (< 100). We discuss applications and show how to adapt our algorithms to dynamic networks such as the Benes network, the interconnection network used in the Meiko CS-2.

1 Introduction

The Multi-Message Multicasting (MM_C) problem over an n processor static network consists of finding a communication schedule with least total communication time for multicasting a 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 are routed through the links of N. The objective is to determine when each of these messages is to be transmitted so that all of the communications can be carried in the least total amount of time.

Routing in the complete static network (there are bidirectional links between every pair of processors), is the simplest and most flexible, when compared to other static networks with restricted structure like rings, mesh, star, binary trees, hypercube, cube connected cycles, shuffle exchange, etc., and dynamic networks, like Omega Networks, Benes Networks, Fat Trees, etc. The minimum total communication time for the MM_C problem is an obvious lower bound for the total communication time of the corresponding problem on any restricted communication network. But, most interesting, the MM_C for dynamic 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 replicators) is not that different, in the sense that the number of communication phases in these dynamic networks is twice of that in the complete network. This is because each communication phase in the complete network can be translated into two communication phases. In the first phase data is replicated and transmitted to other processors, and in

the second phase data is distributed to the appropriate processors ([13], [14], and [16]). One may reduce the translation process to a single step, by increasing the number of network switches about 50% ([13], [14], and [16]). Multiprocessor systems based on Benes networks include the IBM GF11 machine [1], and the Meiko CS-2. The two stage translation process can be used in the Meiko CS-2 computer system and any multimessage multicasting schedule can be realized by using basic synchronization primitives. 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 a variety of dynamic networks.

Formally, processor P_i needs to multicast s_i messages, each requiring one time unit to reach any 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 may receive. We define the *degree* of a problem instance as $d = \mathbf{max}\{s_i, r_i\}$, 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., $s_1 = 3, s_2 = 4$, and $s_3 = 2$). The destinations of all of these messages is: $T_{1,1} = \{2\}, T_{1,2} = \{3\}, T_{1,3} = \{2,3\}, T_{2,1} = \{1\}, T_{2,2} = \{1\}, T_{2,3} = \{3\}, T_{2,4} = \{1,3\}, T_{3,1} = \{1,2\}, T_{3,2} = \{2\}$. In this case $r_1 = 4$, $r_2 = 4$, and $r_3 = 4$.

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 has 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 shown in Figure 1 as a directed multi-graph.

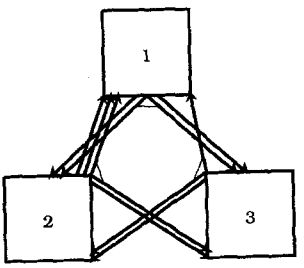

Fig. 1. Directed Multi-Graph Representation for Example 1. The thin line joins all the edges (branches) in the same bundle

The communications allowed must satisfy two restrictions:

1.- During each time unit each processor may transmit one message, but such message can be multicast 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 each message in one or more stages. I.e., each set $T_{i,j}$ can be partitioned into subsets, and each of these subsets is transmitted at a different time. Of course, this does not prevent one from sending a message to all its destinations at the same time. Restricting each message to be transmitted to all of its destinations at the same time increases the total communication time, and in some cases all feasible communication schedules have a total communication time that cannot be bounded (above) by any function $f(d)$. This is why it is important to send each message at different times.

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

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 latest time at which there is a communication which is equal to the number of communication modes in schedule S, and 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 bound we use this simple measure. Another reason for this is that load balancing procedures executed prior to the multicasting require a simple objective function in terms of the problem instance it generates.

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.

In Section 2 we present an efficient algorithm to construct for any degree d problem instance a communication schedule with total communication time at most d^2. Gonzalez [7] has found 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 and the

number of processors is huge. Since this environment is not likely in the near future, we study in subsequent sections important subproblems of the MM_C problems that are likely to arise in practice.

The *basic multicasting problem* (BM_C) is the degree $d = 1$ MM_C problem. The BM_C problem can be trivially solved by sending all the messages at time zero. There will be no conflicts because $d = 1$, i.e., each processor must send at most one message and receive at most one message. When a set of processors is connected via a dynamic network whose basic switches allow replication (input lines may be replicated to several output lines), the basic multicast problem can again be solved in two stages: the replication step followed by the distribution step ([13], [16], [14]).

Let us now consider the case when each message has fixed fan-out k. When $k = 1$ (multimessage unicasting problem MU_C), our problem has been reduced to the Openshop Preemptive Scheduling problem [7] which can be solved in polynomial time [8]. In this case, each degree d problem instance has a d color optimal coloration. 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 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, Garey, Johnson and LaPaugh [2] studied a version of the multi-message unicasting problem when messages have different lengths, each processor can send (receive) $\alpha(P_i) \geq 1$ ($\beta(P_i) \geq 1$) messages simultaneously, and messages are transmitted without interruption (nonpreemptive mode). Whitehead [18] 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 ([4], [5], [3]), Hajek and Sasaki [11], Gopal, Bongiovanni, Bonuccelli, Tang, and Wong [9]. Some of these papers considered the case when the input and output units are interchangeable (can send or receive messages). Rivera-Vega, Varadarajan and Navathe [15] studied, the file transferring problem, a version the multi-message unicasting problem for the complete network when every vertex can send (receive) as many messages as the number of outgoing (incoming) links, all messages have the same length and take one unit of time to move along any link. Our MM_C problem is closest to the Meiko CS-2 communication model, and it involves multicasting rather than just unicasting.

The MM_C problem is significantly harder than the MU_C. Even when $k = 2$ the decision version of the MM_C problem is NP-complete [7]. Gonzalez [7] developed an $O(nd^{2.5})$ time algorithm to construct a communication schedule with total communication time at most $2d - 1$ for every n processor instance of the MM_C with fan-out $k = 2$.

In section 3 we present an algorithm to construct for any problem instance of degree d and fan-out k a communication schedule with total communication time

at most $qd+k^{\frac{1}{q}}(d-1)$, for any integer $q \geq 2$. The time complexity bound for our algorithm is $O(n(d(q+k^{\frac{1}{q}}))^q)$. Our main result is a linear time approximation algorithm with a smaller approximation bound for small values of k (< 100), these are the problem instances most likely to arise in practice.

Multimessage multicasting arises in many applications. Suppose that we have a sparse system of linear equations to be solved via an iterative method (e.g., a Jacobi-like procedure). 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 the x_is and $f_i()$s to the processors. Effective 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 to the set of processors that need them. The strategy is to compute $X(1)$, then perform the multimessage multicasting, then compute $X(2)$, and so on. The same communication schedule can be used at each iteration. Our approximation bounds are in terms of the lower bound d. This facilitates the placement procedure since it seeks a placement that induces a multimessage multicasting problem with minimum density d and small fan-out. Other applications include solution of non-linear equations, and most dynamic programming procedures, since all of the multicasting information depends only on the initial placement, which is determined a priori.

2 General Approximation Bound for the MM_C Problem

We show how to construct a communication schedule with total communication time at most d^2 for every degree d problem instance. Gonzalez [7] has shown that the bound of d^2 is tight in the sense that there are degree d problem instances such that all their communication schedules have total communication time at least d^2. For brevity we do not include this lower bound. It is important to note that the bound of d^2 arises in problem instances with huge fan-out and a huge number of processors. This is why we study in subsequent sections approximation algorithms for problem instances with restricted fan-out, which are the problem instances that are likely to arise in practice.

Let P be any n processor instance of the MM_C problem of degree d. The set of d^2 colors is $\{(i,j)|1 \leq i \leq d \text{ and } 1 \leq j \leq d\}$. Now order the incoming edges to each vertex, and order all the bundles emanating from each vertex. Assign color (i,j) to edge $e = \{p,q\}$ if e belongs to the i^{th} bundle emanating form vertex p, and e is the j^{th} incoming edge to vertex q.

Theorem 1. *The informal algorithm described above generates a communication schedule with total communication time at most d^2 for every degree d instance of the MM_C problem. Furthermore, the algorithm can be implemented to take linear time with respect to the number of edges in the multi-graph.*

Proof. The proof follows from the observation that edges emanating from the same processor belonging to different bundles are colored with different colors, and all the incoming edges to a node are colored with different colors. The total number of colors is d^2. It is simple to show that the time complexity bound for the algorithm is linear with respect to the input size.

3 Approximating the MM_C with Fan-Out $k \geq 3$

Problem instances of degree $d = 1$ can always be colored with one color; the ones of degree $d = 2$ and $k = 3$ can always be colored with four colors; and similar results can be obtained for problems with fixed degree d and fan-out k. For brevity we do not prove these special cases.

3.1 Crude Approximations

Let us now consider some simple approximation algorithms for our problem. The algorithms color all edges emanating from P_1, $P_2, \ldots P_{j-1}$. With respect to this partial recoloration we define the following terms. Each branch emanating from P_j 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.

A coloration in which every message is colored with exactly one color may require as many as $d + k(d-1)$ colors. The reason is that each branch has $d-1$ t_{j-1}-forbidden colors, and none of the t_{j-1}-forbidden colors in a branch can be used to color the corresponding bundle. Therefore, there can be $k(d-1)$ t_{j-1}-forbidden colors that cannot be used in the bundle. Since there are at most d bundles emanating from a processor P_j, and every bundle is assigned one color, then $d + k(d-1)$ colors are sufficient to color all the bundles emanating from processor P_j, and hence the multigraph.

The above upper bound can be decreased substantially by assigning up to two colors per message (bundle). Again, each branch has $d-1$ t_{j-1}-forbidden colors. But, two colors that are not t_{j-1}-forbidden in the same branch of a bundle can be used to color that bundle. For example, if the forbidden colors in the branches are {1,2,3}, {2,3,4} and {2,3,4}, respectively, one can color the first branch with color four, and the other two with color one. So the question is: What is the largest number of t_{j-1}-forbidden colors in a bundle such that no two of them can be used to color the bundle? For $k = 3$ and $d = 7$ it is nine. The t_{j-1}-forbidden colors in the three branches are: $\{1,2,4,5,7,8\}$, $\{1,3,4,6,7,9\}$, and $\{2,3,5,6,8,9\}$. Note that no two of the nine colors can color completely the bundle. We have established that the largest number of t_{j-1}-forbidden colors in a bundle such that no two of them can color completely the bundle is $d-1$ for $k = 2$, about $1.5(d-1)$ for $k = 3$, etc.

We can restate this problem in graph theoretic terms as follows. Find the complete graph with the largest number of vertices such that all its edges are covered by k cliques of size $d-1$. The vertices represent the colors, the cliques

the t_{j-1} − *forbidden* colors, and the number of cliques represent k. By simple counting arguments, the maximum number is less than $\sqrt{k}(d-1)$. Therefore, all the bundles emanating from processor P_j (and therefore the multigraph) can be colored with $2d+\sqrt{k}(d-1)$ colors, the $\sqrt{k}(d-1)$ colors are the t_{j-1}-forbidden colors, and the $2d$ colors are the one used in the coloration. One can easily prove smaller bounds, $3d-1$ for $k=2$; is about $3.5d-1.5$ for $k=3$; etc (see [7]).

The obvious generalization is to use q colors instead of two for each bundle. To find th number of colors needed in this case we need to generalize graphs to hypergraphs. A q-hypergraph consists of a set of vertices and a set of q-hyperedges, where a q-hyperedge is just a subset of q vertices. Clearly, a 2-hypergraph is just a graph. So the previous graph problem becomes, find the complete q-hypergraph with the largest number of vertices such that all its q-hyperedges are covered by k q-hypercliques of size $d-1$. By simple counting arguments, the maximum number of vertices is less than $k^{\frac{1}{q}}(d-1)$. Therefore, all the bundles emanating from processor P_j (and therefore the multigraph) can be colored with $qd+k^{\frac{1}{q}}(d-1)$ colors. The time complexity bound in this case is $O(n(d(q+k^{\frac{1}{q}}))^q)$, by trying all subsets of colors of size q. The algorithm will color with at most q colors all the bundles emanating from each processor. We now state our result without its proof.

Theorem 2. *For every instance of the MM_C problem with fan-out $k \geq 3$, the informal algorithm generates in $O(n(d(q+k^{\frac{1}{q}}))^q)$ time a schedule with total communication time $qd+k^{\frac{1}{q}}(d-1)$.*

Gonzalez [7] has developed a linear time algorithm, with respect to the input length, to generate a valid coloration for the above case. For brevity we do not include these result in this paper.

In what follows we present another procedure and carry out a much **sharper** analysis. Table 1 has the coefficient for d for different methods. The ones labeled "simple" are for the "crude" methods. The "involved (2c)" is the method we discuss in the next subsections. The other methods appear in [7] and for brevity are not described in this paper.

3.2 Sharper Approximation

We present an approximation algorithm and carry out a much **sharper and involved** analysis of its performance. The input to our algorithm is a multigraph G, and integers h and l that restrict the color selection process ($k > l > h \geq 1$). Note that k and d can be extracted from the graph. The algorithm colors the edges emanating out of P_1, then P_2, and so on. When considering processor P_j, a color is selected from each bundle from the set C_i with smallest index, and then the existence of a second color for the remaining branches is guaranteed by just having enough colors available. Before we present our results we define some useful terms.

At the beginning of the j^{th} iteration the algorithm has colored all the branches emanating from processors $P_1, P_2, \ldots, P_{j-1}$ and it is ready to begin coloring all

Table 1. Number of Colors For The Different Methods.

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

the branches emanating from P_j. For $0 \leq i \leq k$, let C_i^J be the set of colors that are t_{j-1}-forbidden in exactly i branches of bundle J. Let $c_i^J = |C_i^J|$. (When the set J is understood, we will use c_i for c_i^J, and C_i for C_i^J.) 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 $\sum_{i=1}^{k} iC_i^J \leq (d-1)k$ for each bundle J emanating from P_j. Clearly, all the branches of bundle J can be colored with any of the colors in C_0^J. Also, one can color all the branches of bundle J with two colors, $a \in C_i^J$ and $b \in C_j^J$ provided that colors a and b are not t_{j-1}-forbidden in the same branch of bundle J and have not been used to color another branch emanating from processor P_j. Just after coloring a subset of branches of a bundle 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.

The input to the algorithm consists of G (the multi-graph), and h and l (to restrict the selection of colors). Note that k (the fan out), and d (the degree) can be computed from G. Later on give the specific values for which the algorithm is defined.

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

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

Our algorithm requires that $d \geq \frac{2l+2h^2}{h^2+3h-2}$, $k \geq L$, $k > l > h \geq 1$ and $d > 4$, which we will show later on is not a limiting factor. We begin by establishing in Lemma 3 that $L \leq R$, which we state without its proof. This fact will be used to partition in two cases the set of values for which our algorithm is defined.

Lemma 3. *If* $d \geq \frac{2l+2h^2}{h(h+3)}$ *then* $L \leq R$.

In Table 2 we define equations eq.(0), ..., eq.$(h+1)$ that are used by the algorithm and are necessary for the correctness proof.

Procedure Coloring, whose input consists of the multi-graph G, and integers h and l, is given below. For the set of valid inputs, defined above, procedure

Table 2. Equations eq.(0), ..., eq.$(h+1)$

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

computes the maximum number of colors needed (Δ) and a coloration for G with at most Δ colors.

Procedure Coloring (G, h, l)
/* Note that k, d, L, and R can be easily computed from G */
/* Procedure is defined for $d \geq \frac{2l+2h^2}{h(h+3}$, $k \geq L$, $k > l > h \geq 1$, 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
for each processor P_j **do**
for each bundle J emanating from processor P_j **do**
compute $C_0^J, C_1^J, C_2^J, \ldots C_k^J$;
let p_J be the smallest integer such that equation eq.(p_J) holds;
let $q_J =$ **min** $\{p_J, h\}$;
let $r_J = p_J$ if $0 \leq p_J \leq h$ and $r_J = l$ otherwise;
endfor
/* Color a subset of edges emanating from each bundle of P_j */
for each uncolored bundle J of P_j **do**;
color as many branches of bundle J with an s_j-free color in $C_0^J, C_1^J, \ldots, C_{q_J}^J$;
/* Color the remaining uncolored edges emanating from P_j */
for each partially colored bundle J of P_j with uncolored branches **do**;
color all uncolored branches of J with an s_j-free color in $C_0^J, C_1^J, \ldots, C_{r_J}^J$;
endfor;
end of Procedure Coloring

To establish that Procedure Coloration generates a valid coloration for the cases it is defined is difficult. At this point the readers might feel there are a large number of cases for which our algorithm is not defined, but we have established that these cases can be ignored because the corresponding graph G with other values for h and l that are valid for our algorithm and requires a smaller Δ.

In other words, the cases that are omitted by our algorithm do not enhance the overall performance of our algorithm. In Theorem 8 we establish that our algorithm generates valid colorations and that it takes linear time with respect to the input length. The proof of this theorem is based on Lemmas 6 and 7 that are used to establish that two colors can always be selected from the appropriate sets to color all the branches from the bundles that could not be colored with exactly one color. Lemma 5 is used in the proof of Lemmas 6 and 7, and requires Lemma 4. For brevity we just list the lemmas without their involved proofs. The proof of Theorem 8 shows the need for the lemmas.

Lemma 4. *If* $d \geq \frac{2l+2h^2}{h(h+3)}$ *then* $(h+1)^2 - \frac{h^2}{d-1} + \frac{h+1}{d-1} \leq R$

Lemma 5. *If* $d \geq \frac{2l+2h^2}{h(h+3)}$ *and* $k \geq L$, *then the value for* Δ *defined by Procedure Coloration is at greater than or equal to* $(h+2)d - 2h$.

Lemma 6. *If* $k \geq L$ *and* $d \geq \frac{2l+2h^2}{h(h+3)}$, *then at the beginning of the* j^{th} *iteration of Procedure Coloring each bundle* J *emanating from processor* P_j *satisfies* $\sum_{i=0}^{h} c_i^J \geq d$.

Lemma 7. *If* $d \geq \frac{2l+2h^2}{h(h+3)}$ *and* $L \leq k$, *then at the beginning of the* j^{th} *iteration of Procedure Coloring each bundle* J *emanating from processor* P_j *satisfies at least one of the inequalities eq.*(j), *for* $0 \leq j \leq h+1$, *holds.*

Theorem 8. *For every instance of the* MM_C *problem with fan-out* $k \geq 2$, $d \geq \frac{2l+2h^2}{h(h^2+3)}$ *and* $L \leq k$, *Procedure Coloring generates a communication schedule with total communication equal to the value of* Δ *computed by the algorithm. The time complexity of the procedure is linear with respect to the input size.*

Proof. First we prove that Procedure Coloration colors all the edges in the graph with a number of colors equal to Δ, as computed 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 7 we know that at least one of the equations eq.(i) for $0 \leq i \leq h+1$ holds for each bundle emanating from P_j. Therefore, all the p_J values are integers in the range [0,h+1], and all the q_J 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 $C_0^J, C_1^J, \ldots C_{q_j}^J$. We prove this by showing that $\sum_{i=0}^{q_J} c_i^J \geq d$, since this fact guarantees that one unique s-free color in $C_0^J, C_1^J, \ldots C_{q_j}^J$ for each bundle J can be selected in the first loop to color a nonempty subset of edges emanating out of each bundle. As we established before, $q_J \leq h$. If $q_J = h$ then by Lemma 6 it follows that $\sum_{i=0}^{q_J} c_i^J \geq d$. On the other hand, if $q_J < h$ then by definition of q_J and Lemma 7 we know that $eq.(q_J)$ holds. This implies that either $c_0 \geq d$ or $\sum_{i=0}^{q_J} c_i \geq (q_J+2)d - 2q_J$. Since $d > 2$, it then follows that $\sum_{i=0}^{q_J} c_i^J \geq d$. Therefore, in the first loop one can select unique s-free color in $C_0^J, C_1^J, \ldots C_{q_j}^J$ for each bundle J to color a nonempty subset of edges emanating out of each bundle.

We now claim that at each iteration in the second loop one can select unique colors to color the remaining uncolored branches of each bundle. Remember that $\sum_{i=0}^{r_J} c_i \geq (r_J+2)d-2r_J$. The number of colors that were t_{j-1}-forbidden in the same branch as the color selected in the previous loop is at most $(d-2)\cdot q_J$, and the maximum number of colors used during both loops is at most $2d-1$. It follows that the colors that one can use to color the remaining branches are at least $(r_J+2)d-2r_J-(d-2)\cdot q_J-2d+1$. This is equivalent to $(d-2)(r_J-q_J)+1$. Since $d > 2$ and $r_J \geq q_J$, 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 all the steps take time $O(ndk)$, and can be implemented to take linear time with respect to the input length.

4 Discussion

For the case of $k = 3$ the approximation bound can be shown to be $\frac{5d-4}{2}$ and one can establish that any algorithm that colors the bundles emanating form each vertex at a time without recoloration must use at least $\frac{7d-3}{3}$ colors. As we mentioned before, for brevity we just presented some of the simpler approximation algorithms we have developed. It is worth noting that the proofs of all the lemmas in the previous subsection can be proved by a Symbolic Manipulator System such as Mathematica after adding several macros and functions. The need to use symbolic manipulators arose from the complexity of the expressions that need to be handled.

The MM_C problem can be viewed as the generalization of the multigraph edge coloration (EC) problem [17], where the edges are directed, and bundled in groups. Vizing's [17] approximation algorithm for the EC problem colors the edges one at a time with one of the $1.5d$ available colors, where d is the degree of the multigraph. If necessary the algorithm recolors some edges in order to color an edge. But, the necessary backtracking is limited. Recoloration in our problem is harder, because at each node there may be many edges that need to be recolored, rather than just two as in Vizing's algorithm. Our currently best approximation algorithm allows limited recoloration.

References

1. G. S. Almasi, and A. Gottlieb, *Highly Parallel Computing*, The Benjamin/Cummings Publishing Co., Inc., New York, 1994.
2. E. G. Coffman, Jr, M. R. Garey, D. S. Johnson, and A. S. LaPaugh, Scheduling File Transfers in Distributed Networks, *SIAM Journal on Computing*, 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. H.-A. Choi, and S. L. Hakimi, Scheduling File Transfers for Trees and Odd Cycles, *SIAM Journal on Computing*, Vol. 16, No. 1, February 1987, pp. 162 – 168.

5. H.-A. Choi, and S. L. Hakimi, "Data Transfers in Networks with Transceivers," *Networks*, Vol. 17, (1987), pp. 393 – 421.
6. T. F. Gonzalez, "Unit Execution Time Shop Problems, " *Mathematics of Operations Research,"* Vol. 7, No. 1, February 1982, pp. 57 - 66.
7. T. F. Gonzalez, "Multimessage Multicasting in Networks, " UCSB Technical Report, (in preparation).
8. T. F. Gonzalez, and S. Sahni, Open Shop Scheduling to Minimize Finish Time, *Journal of the Association for Computing Machinery*, Vol. 23, No. 4, October 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. A J. Hopcroft, and R. M. Karp, An $n^{2.5}$ Algorithm for Maximum Matchings in Bipartite Graphs, *SIAM J. Computing,* (1973), pp. 225 – 231.
11. B. Hajek, and G. Sasaki, Link Scheduling in Polynomial Time, *IEEE Transactions on Information Theory*, Vol. 34, No. 5, Sept. 1988, pp. 910 – 917.
12. I. Holyer, The NP-completeness of Edge-Coloring, *SIAM J. Comput.*, 11 (1982), pp. 117 – 129.
13. T. T. Lee, Non-blocking Copy Networks for Multicast Packet Switching, *IEEE J. Selected Areas of Communication*, Vol. 6, No 9, Dec. 1988, pp. 1455 – 1467.
14. S. C. Liew, A General Packet Replication Scheme for Multicasting in Interconnection Networks, *Proceedings IEEE INFOCOM '95*, Vol.1 (1995), pp. 394 – 401.
15. P. I. Rivera-Vega, R, Varadarajan, and S. B. Navathe, "Scheduling File Transfers in Fully Connected Networks," *Networks*, Vol. 22, (1992), pp. 563 – 588.
16. J. S. Turner, A Practical Version of Lee's Multicast Switch Architecture, *IEEE Transactions on Communications*, Vol. 41, No 8, Aug. 1993, pp. 1166 – 1169.
17. V. G. Vizing, On an Estimate of the Chromatic Class of a p-graph, *Diskret. Analiz.*, 3 (1964), pp. 25 – 30 (In Russian).
18. J. Whitehead, The Complexity of File Transfer Scheduling with Forwarding, *SIAM Journal on Computing* Vol. 19, No 2, April 1990, pp. 222 – 245.