36 Message Dissemination Using Modern Communication Primitives

Teofilo F. Gonzalez
University of California

36.1 Introduction

In this chapter, we discuss algorithms, complexity issues, and applications for message dissemination problems defined over networks based on modern communication primitives. More specifically, we discuss the problem of disseminating messages in parallel and distributed systems under the multicasting communication mode. These problems arise while executing in a parallel computing environment iterative methods for solving scientific computation applications, dynamic programming procedures, sorting algorithms, sparse matrix multiplication, discrete Fourier transform, and so forth. These problems also arise when disseminating information over sensor and ad hoc wireless networks.

The goal of this chapter is to collect scattered research results developed during the last decade to establish that the multicasting communication environment is a powerful communication primitive that allows for solutions that are considerable better than those achievable under the telephone (or one-to-one) communication environment. The multicasting communication environment has been available for quite some time in parallel computing systems. We also establish that within the multicasting communication mode forwarding plays an important role by allowing solutions that are considerable better than when restricting to direct communications, even when the communication load is balanced and the network is completely connected. Offline communication scheduling allows for considerably better solutions over online scheduling. Though online scheduling provides added flexibility and it is applicable to a larger set of scenarios.

The goal of parallel and distributed systems is to speedup the processing of programs by a factor proportional to the number of processing elements. To accomplish this goal a program must be partitioned into tasks, and the communications that must take place between these tasks must be identified to ensure a correct execution of the program. To achieve high performance one must assign each task to a processing unit (statically or dynamically) and develop communication programs to perform all the intertask communications efficiently. Efficiency depends on the algorithms used to route messages to their destinations, which is a function of the communication environment. Given a communication environment and a set of messages that need to be exchanged, the message dissemination problem is to find a schedule to transmit all the messages in the least total number of communication rounds. Generating an optimal communication schedule, that is, one with the least total number of communication rounds (or minimum total communication time), for our message dissemination problems over a wide range of communication environments is an NP-hard problem. To cope with intractability efficient message dissemination approximation algorithms have been developed for different types of communication environments. Let us begin by discussing the different components of our message dissemination problems: message communication patterns (the communications that must take place), and communication environment. The communication environment consists of the communication network (the direct communications allowed for each processor), primitive operations (the basic communication operations allowed by the system), and the communication model (possible operations during each communication round or step).

Message Communication Patterns: We refer to one of the most general message communication problem as the *multimessage multicasting* problem. In this problem every processor needs to communicate a set of messages, each to a set of processors. We use *multimessage* to mean that each processor sends a set of messages, and *multicasting* means that each message is sent to a set of processors. The extreme cases of multicasting is when messages are sent to only one destination (called *unicasting*) and to all possible destinations (called *broadcasting*). A more general version of the problem is when the messages have different lengths, but it is "equivalent" to the multimessage multicasting problem when the messages are partitioned into packets. A restricted version of the multimessage multicasting is the multimessage unicasting problem, in which all the messages have exactly one destination. Another restricted form of the problem is the *gossiping* problem where each processor sends only one message, but each message is sent to all the other processors. Further restrictions include the single message broadcasting, and single message multicasting problems. For some communication environments even these simple versions of the message dissemination problem are NP-complete.

Network: The complete static communication network (there are bidirectional links between every pair of processors) is the simplest and most flexible when compared to other static networks (or simply *networks*) with restricted structure like rings, mesh, star, binary trees, hypercube, cube connected cycles, shuffle exchange, and so forth, and dynamic networks (or multistage interconnection networks), like Omega Networks, Benes Networks, Fat Trees, and so forth. The minimum number of communications rounds needed to carry out any communication pattern over a complete network is an obvious lower bound for the corresponding problem over any communication network.

Primitive Operations: Under the classical *telephone* communication primitive when a processor sends a message it is sent to just one of its neighbor processors at a time. In order to send a message to k destinations one needs to send a copy of the message to one destination at a time. This takes k communication rounds. When *message forwarding* is allowed the processor sending the message may send fewer than k messages, but one or more of the intermediate processors that receives the message must forward it to other destinations.

The *multicasting* communication primitive allows a processor to send a message during each communication round to a subset of its adjacent processors. This communication mode allows multidestination messages to be transmitted faster, and it has been available for quite some time in parallel computing systems. It is a natural communication mode in wireless networks. In our communication problems one may use multicasting efficiently by identifying messages with disjoint origins and destinations so that they may be transmitted concurrently. This is not always possible; however, to deal with problem instances that lack concurrency one may conveniently partition message destinations into groups and messages will be transmitted to each group of destinations during different communication rounds. We refer to this operation as *splitting message destinations*. Forwarding is also available when using the multicasting communication primitive. In this chapter when forwarding is allowed it is specified explicitly. There are situations when forwarding is not allowed. For example, forwarding is expensive in all optical networks because every forwarding operation implies that an optical signal is transformed into an electrical one and then to an optical one. These are expensive operations. For many years there have been discussions about the multicasting communication mode over the Internet, but it is only partially operational and the part that is operational is not fully utilized.

Communication Model: The *single port communication mode* allows every processor to send at most one message and receive at most one message during each communication round. The *multiport communication mode* is a more general model that allows a processor to send/receive more than one message per communication round provided that only one message is transmitted per communication link. The advantage of this more general model is that all the communications may be carried out faster, but there is an additional cost associated with the hardware equipment needed to achieve this communication concurrency. In what follows we discuss only the single port communication mode. A relatively inexpensive way to reduce the total communication time (which is just the total number of communication rounds) is to add buffers at the receiving end of each processor. Additional hardware is needed for the buffers and to control their behavior. However, the buffers make a single port communication mode behave more like a multiport communication at a fraction of the cost.

The problem is to develop algorithms that generate near-optimal solutions to message dissemination problems under the multicasting communication environment. These problems have wide applicability in all sorts of systems. The algorithms we have developed provide the initial solution that may then be fine tuned as new technological innovations take place.

Executing in a parallel processing environment iterative methods for large scientific computations give rise to multimessage multicasting problems. A simple example is solving large sparse system of linear equations through stationary iterative methods [1]. Other applications arise when dynamic programming procedures are being executed in a parallel or distributed computing environment. Sorting, matrix multiplication, discrete Fourier transform, and so forth [2, 3] are applications with significant message dissemination operations. Multicasting information over a b-channel ad hoc wireless communication network is another important application in information systems. The multicasting communication environment arises naturally in sensor and ad hoc wireless networks. Because the cost of deployment is decreasing rapidly, these types of networks are finding new applications. Ad hoc wireless networks are suited for many different scenarios including situations where wired Internet or Intranet is not practical or possible.

The solution of sparse systems of linear equations through an iterative method in a parallel computing environment gives rise to one of the simplest forms of multimessage multicasting. Given vector $X[0] = x_1[0], x_2[0], \ldots, x_n[0]$, the vectors $X[t]$, for $t = 1, 2, \ldots$, need to be computed as $x_i[t+1] = f_i(X[t])$. However, f_i includes very few terms, because the system is sparse. In a parallel computing environment, the x_is are assigned to the processors where the function f_i computes the new values. The x_is computed at each iteration need to be transmitted to the processors that require them to evaluate the functions f_i at the next iteration. The computation starts by transmitting the $X[0]$ elements to the appropriate processors and then use them to compute $X[1]$. Then $X[1]$ is transmitted and used to compute $X[2]$, and so on. This process repeats until the difference between the old and new values are smaller than some given tolerance, or a certain number of iterations have taken place. The same communication schedule is used at each iteration. The schedule is computed offline once the placement of the x_is has been determined. When the computation and communication loads are balanced, full speedups are possible provided the computation and communication time can be overlapped. The solution to the linear equations problem is more dynamic and converges faster when one uses the newly computed values as soon as they become available. A very important problem in this whole process is the placement of the x_is, but for brevity it is not discussed in this chapter.

The offline multimessage multicasting problem is fully justified by the above application. The "distributed" or "nearly online" multimessage multicasting problem is an interesting variation. Message destinations are not known until a portion of a task has been executed and this information is available only locally at the processor executing the task. At a given synchronization point all the processors start performing their communication steps. The communication schedule needs to be constructed online. Globalization of the information and the time required to compute the communication schedule need to be taken into consideration when the performance of the algorithm is being evaluated. Message dissemination problems are solved offline, unless we mention explicitly that they are distributed or online versions of the problem.

A class of parallel computers use dynamic networks for communications. These computers include the now extinct Meiko CS-2 and the IBM GF11 machines [4] that use a variation of the Benes communication network. The computer chips implementing the basic Meiko CS-2 switches are still operational connecting processors in distributed environments. A *pr-dynamic network* [1] is a dynamic network that performs in one communication round any single permutation communication step, that is, every processor sends and receives one message, and the "switches" can replicate its input data to any subset of its output ports. The most interesting property of pr-dynamic networks is that any communication pattern over a complete network can be automatically converted into an equivalent communication pattern for any pr-dynamic network that carries out all the communications in just two communication rounds. In the first round, data is replicated and in the second one, data is distributed to the appropriate location [5–7]. Communication networks that are about 50% larger can reduce the two communication rounds to one.

In our analysis we concentrate on the multimessage multicasting over complete undirected networks. This version of the problem has a simple structure that is simple to analyze. Furthermore, any communication schedule for a complete undirected network can be automatically transformed into one for any pr-dynamic network. This transformation doubles the total communication time. However, for the best of our algorithms, this communication step does not introduce any additional communication rounds. Specifically, when multicasting operations have adjacent processors as destinations, or the destinations of any two messages transmitted currently are not interleaving (if a message has destination processors i and j for $i < j$, then no other message may have destination k, for $i < k < j$ during the communication round), can be implementation in a pr-dynamic network in a single communication round. The schedules constructed by the best of our forwarding algorithms satisfy the above property. Another reason for concentrating our study on the multimessage multicasting over completely connected networks is that it also models a fixed set of processors over an optical communication ring.

In an n processor system connected through a Benes network messages may need to traverse $\Omega(\log n)$ switches to reach their destination. Our communication model refers to this path as a single communication step that takes one communication round. This simplification is acceptable because the amount of time needed to prepare a message for transmission (moving the data, and setting up the routing table) is more than the message transmission time through the switches (which are implemented in silicon chips). So the problem is simply minimizing the total number of communication rounds required for the transmission of all messages, even when n is not too large. For large n, messages may interfere with the transmission of other messages and cause the total communication time to be much larger than the time to set up all the messages. Strictly speaking, our analysis is not scalable, but our assumptions are acceptable for systems with up to several hundreds of millions of processors. We do not expect to have networks with more than this number of processors in the near future. Processors interconnected through wrap-around pr-networks allow for some communication patterns to traverse only a constant number of switches to reach their destination. Algorithms may be designed to take advantage of this, but most of the time it is impossible to gain from the implementation of this strategy. Since there is a gain only in relatively few cases, we do not take into consideration this level of detail in our model. Our assumptions are reasonable ones that simplify the analysis and allow us to concentrate on the most important communication issues.

As we have discussed earlier, our analysis has direct applications from sensor networks to parallel and distributed computing. Other applications in high-performance communication systems include voice and video conferencing, operations on massive distributed data, scientific applications and visualization, high-performance supercomputing, medical imaging, and so forth. The delivery of multidestination messages will continue to increase in the future and so our problems will find more applications with time. This work is laying the foundations over which applications can run smoothly.

When the communications are restricted to the telephone mode, the best possible schedules have total communication time that is not bounded by any function in terms of d. For example, when there is only one message to be delivered to n destinations, any schedule requires $\log n$ communication rounds even when the network is fully connected. However, under the multicasting communication primitive it requires only one communication round. The multicasting communication primitive delivers performance that is not attainable under the telephone communication mode. This is an important consequence of the research accomplishments reported in this chapter.

The distributed version of the multimessage multicasting problem is much more general than the offline versions, but their communication schedules have $\Omega(d + \log n)$ expected number of communication rounds. One can always construct schedules for the multimessage multicasting problem with a number of communication rounds of at most d^2, and just $2d$ when forwarding is allowed. Therefore, there is a clear advantage when we have knowledge ahead of time of all the communication requirements. Forwarding is another important factor in reducing the total communication time. These are important consequences of the research accomplishments reported in this chapter.

In Section 36.2 we formally define our problems and discuss related work. Then in Section 36.3 we discuss the multimessage multicasting problem. We establish NP-completeness results, provide upper and lowers bounds for the communication time, and discuss restricted versions as well as generalizations of the problem. Section 36.4 discusses similar issues, but for the multimessage multicasting problem with forwarding. Communication schedules with significantly better communication time are achievable when forwarding is allowed. The distributed or online version of the multimessage multicasting problem with forwarding is discussed in Section 36.5.

Section 36.6 discusses the *gossiping* communication problem under the multicasting communication mode. From our definitions we know gossiping is a restricted version of multimessage multicasting; however, we treat these two problems separately because the work on multimessage multicasting has been limited to a class of interconnection architectures, whereas the research results for the gossiping problems apply to all kinds of networks. That is, by restricting the communication patterns provable good solutions can be generated for arbitrary networks.

Section 36.7 considers the multiport communication model in the form of multimessage multicasting for an all-optical multifiber star networks. In this case the messages are carried on different wavelengths in one of several fibers. There is limited switching that permits messages on the same wavelength but different fibers to be interchanged. The effects of limited switching on the total communication time are discussed.

36.2 Definitions and Related Work

The multimessage multicasting communication problem under the single port fully connected network with the multicasting communication primitive is denoted by MM_C. Formally, there is a set of n processors, $P = \{P_1, P_2, \ldots, P_n\}$, interconnected through a complete network (all possible bidirectional links are available). Processors alternate synchronously between computation and communication. During each computation phase, the processors generate a set of messages each of which is to be transmitted to a subset

TABLE 36.1 Source and Destination Processors for the Messages in Example 36.1

Message	*A*	*B*	*C*	*D*	*E*	*F*
Source	1	1	2	2	3	3
Destination	$\{4, 5, 6\}$	$\{7, 8, 9\}$	$\{4, 5, 7, 8\}$	$\{6, 9\}$	$\{5, 6, 8, 9\}$	$\{4, 7\}$

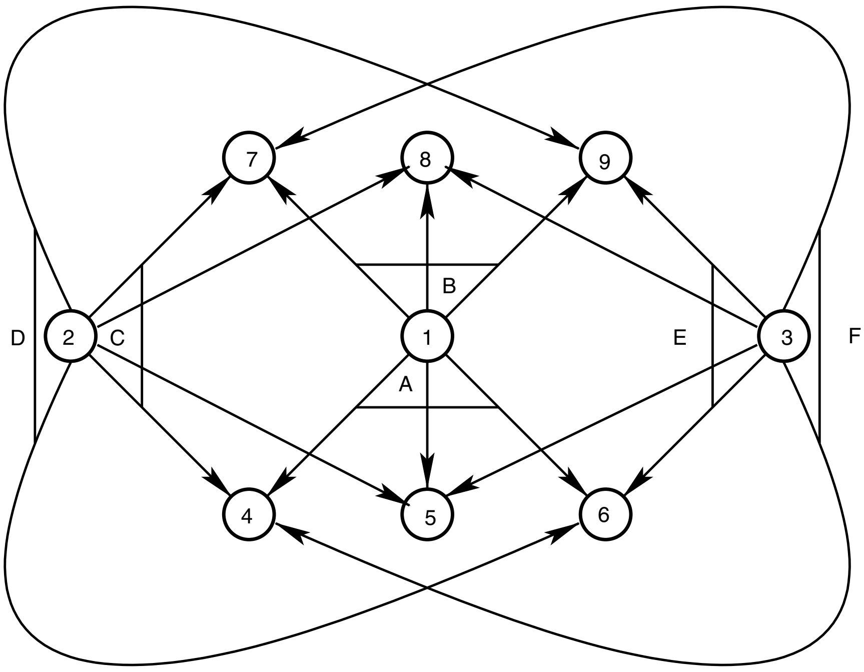

FIGURE 36.1 Directed Multigraph representation for Example 36.1.

of processors (*destinations*). The processor that generates a message is called the *source or originating* processor for the message. All the source messages for P_i form the *hold* set (h_i). By using the message destination information one may compute the messages each processor P_i *needs* to receive (n_i).* The maximum number of messages that any processor needs to send or receive is denoted by $d = \mathbf{max}\{|\, h_i \,|, |\, n_i \,|\}$. Example 36.1 gives an instance of the MM_C problem.

Example 36.1 *The number of processors, n, is nine. Six messages need to be disseminated. The source processor and destination processors for the messages are given in Table 36.1. In general all processors need to send and receive messages. For simplicity in this example the first three processors (P_1, P_2, and P_3) just sent messages and processors P_4 through P_9 just receive messages. Every processor sends and receives at most three messages. Therefore, the degree of the instance, denoted by d, is 3.*

There are several ways one may visualize an MM_C problem instance. In the directed multigraph representation, processor P_i is represented by a vertex labeled i and a message is represented by a set of edges or branches originating at the source processor with an edge ending at each destination processor. All the edges associated with a message are *bundled* together. This is represented by drawing a line joining all the directed edges bundled together (see Figure 36.1). Figure 36.1 depicts the directed multigraph

*The n_is do not need to be computed explicitly, but the communication process must deliver to every processor all the messages it needs.

representation for the problem instance given in Example 36.1. Another way to represent the edges associated with a message is by using a directed hyperedge.

The message communications in our complete network must satisfy the restrictions imposed by our communication model given below:

1. During each communication round each processor P_i may multicast a message in its initial hold set h_i (or the current hold set when forwarding is allowed) to a set of processors. The message transmitted remains in the hold set h_i.
2. During each communication round each processor may receive at most one message. A message that processor P_i receives (if any) is added to its hold set h_i and it is available for the next communication round. When two or more messages are sent during the same communication round to the same processor, the processor does not receive any of the messages.* The processors sending the messages will be informed that their message it sent did not reach all its destination. The processors will not know the destinations that were not reached.

When every processor receives all the messages it needs, $n_i \subseteq h_i$ for all i, the communication phase terminates. The total number of communication rounds is referred to as the *total communication time* or simply TCT.

By using the directed graph representation the MM_C problem is an edge coloring problem where the coloring is such that no two edges emanating from the same processor belonging to different bundles are colored identically, and no two edges incident to the same vertex are colored identically. The colors correspond to the communication rounds and the coloring restrictions enforce the communication rules. Our notation is not consistent so we use interchangeably colors, communication rounds or steps, and time; vertices and processors; bundle, branches or edges, and messages. Note that when message forwarding is allowed, the above representation breaks down. One may use edge multicolorings to describe classes of forwarding algorithms [8], but describing an MM_C problem as an edge coloring problem is awkward since forwarding means that an edge is replaced by a sequence of edges whose coloring must satisfy some additional constraints.

From the communication rules it is clear that any message may be transmitted during one or more communication rounds. In other words, a processor may send a message to a subset of its destinations at a given time and to another subset of destinations at another time. We refer to this as *splitting message destinations*. Table 36.2 gives a communication schedule (with message splitting) for Example 36.1 that takes four communication rounds. The i-th row in the table specifies the multicasts to be performed during the i-th communication round. A stricter version of the problem requires every message to be sent to all its destinations during one communication round. We refer to this case as the *unsplittable message destinations*. The problem instance given in Example 36.1 requires five communication rounds when message destinations are unsplittable. The reason for this is that messages originating at the same processor or having at least one common destination cannot be transmitted during the same communication round. Therefore, the only two messages that may be transmitted concurrently are messages D and F, and any schedule for this version of the problem must have TCT at least five. This is not too different from the TCTs of the schedule with splittable message destinations given in Table 36.2. However, there are unsplittable message destination problem instances where all schedules have TCT that cannot be bounded above by any function in terms of d, but, as we shall see later on, all the instances with splittable message destinations have a schedule with TCT at most d^2. Our model allows splittable message destinations because it can significantly improve the TCT and it can be implemented in practice.

We claim that all the communication schedules for the problem instance given in Example 36.1 require at least four communication steps (when splitting message destinations is allowed). The proof of this fact is by contradiction. Suppose there is a schedule with total communication time equal to three. Then either message A or B is transmitted to all their destinations in one time unit. Without loss of generality let's

*This situation does not normally arise when constructing schedules offline, however, it is common in "distributed" or "online" versions of the problem.

TABLE 36.2 Message Transmissions for Schedule S

Communication round	Concurrent communications		
1	A: $1 \rightarrow \{5,6\}$	C: $2 \rightarrow \{7\}$	E: $3 \rightarrow \{8,9\}$
2	B: $1 \rightarrow \{7,8,9\}$	C: $P_2 \rightarrow \{4\}$	E: $3 \rightarrow \{5,6\}$
3	D: $2 \rightarrow \{6,9\}$	F: $3 \rightarrow \{4,7\}$	—
4	C: $2 \rightarrow \{5,8\}$	—	—

TABLE 36.3 Message Transmissions (with Forwarding) for Schedule T

Communication round	Concurrent communications			
1	C: $2 \rightarrow \{4,7\}$	E: $3 \rightarrow \{5,6,8,9\}$	—	—
2	A: $1 \rightarrow \{4,5,6\}$	D: $2 \rightarrow \{9\}$	F: $3 \rightarrow \{7\}$	C: $7 \rightarrow \{8\}$
3	B: $1 \rightarrow \{7,8,9\}$	D: $P_2 \rightarrow \{6\}$	F: $3 \rightarrow \{4\}$	D: $4 \rightarrow \{5\}$

say that it is message B and that the message is transmitted during communication round 1. Then each of the messages C, D, E, and F emanating from processors P_2 and P_3 with destinations P_7, P_8, and P_9 must be transmitted during communication rounds 2 and 3 in one time unit each. But message E cannot be transmitted concurrently with C, D, or F. Also, messages C, D, or F cannot be transmitted concurrently. Therefore, a schedule with TCT at most three does not exist for this problem instance. For this version of the problem all schedules must require at least four communication rounds. However, forwarding allows for communications schedules with TCT equal to three. Table 36.3 gives one such schedule. The only message that is forwarded is message D that is transmitted to processor P_5 through processor P_4. This is possible because the network is fully connected. The difference in the TCT is not significant for the instance given in Example 36.1. However, as we shall see later on there are problem instances that require d^2 TCT without forwarding, but all problem instances have a communication schedule with TCT at most $2d$ when forwarding is allowed.

The maximum number of destinations for messages in a problem instance is called the *fan-out* k. When $k = 1$ each message has at most one destination so the problem is referred to the *multimessage unicasting* MU_C problem. Gonzalez [1] established that this problem corresponds to the minimum makespan openshop preemptive scheduling problem that can be solved in polynomial time. Since all the nonzero tasks have the same length, the schedule constructed by the algorithm in Reference 9 does not have preemptions and, every problem instance of degree d has a schedule with makespan equal to d. Therefore, there is always a schedule with TCT equal to d for the MU_C problem. The makespan openshop preemptive scheduling problem may be viewed as a generalization of the problem of coloring the edges of bipartite multigraphs. There are well-known algorithms that generate optimal colorings for these graphs [10–14]. The fastest of these algorithms is the one by Cole et al. [13]. This algorithm is the best one to solve the MU_C problem when the edge multiplicities are small [14]. However, Gonzalez and Sahni's [9] openshop algorithm is currently the fastest method for the general case. The algorithm given in Reference 13 is for multigraphs so there are multiple edges between pair of nodes. The algorithm in Reference 9, for the openshop problem, represents multiple edges between vertices by a weighted edge. In both of these papers m (or $|E|$) is the total number of edges, but in Reference 13 m is not $O(n^2)$ and may be very large compared to n^2, whereas in Reference 9 it is always less than n^2. One should keep this in mind when comparing these algorithms.

The MU_C problem and its variants have been studied. The basic results include: heuristics, approximation algorithms, polynomial time algorithms for restricted versions of the problem, and NP-completeness results. Coffman et al. [15] present approximation algorithms for a generalization of the MU_C problem where messages have different lengths, but need to be communicated without interruption (preemptions are not allowed), and there are $\gamma(P_i)$ sending or receiving ports per processor. The power of message forwarding was reported by Whitehead [16]. When preemptions are allowed, messages may be transmitted with interruptions, generalizations of the MU_C problem have been considered by

Choi and Hakimi [17, 18], Hajek and Sasaki [19], Gopal et al. [20]. The n-port MU_C problem over complete networks, for transferring files, was studied by Rivera-Vega et al. [21]. A variation of the MU_C problem, called *the message exchange problem*, has been studied by Goldman et al. [22]. The communication model in this version of the problem is asynchronous and it closely corresponds to actual distributed memory machines. Another restricted version of the MU_C problem, called *data migration*, was studied by Hall et al. [23].

The main difference between the above research and the one we discuss in this chapter is that we concentrate on the multicasting communication mode, rather than on the *telephone communication mode.* Previous research on the basis of the multicasting communication mode has concentrated on single messages. Multimessage multicasting results are reported in References 1,8, and 24–35. The initial research by Shen [34] on multimessage multicasting was for n-cube processor systems. The objective function was very general and attempted to minimize the maximum number of hops, amount of traffic, and degree of message multiplexing, so only heuristics could be developed. More recently, different strategies for solving multimessage multicasting problems over all-optical networks were surveyed by Thaker and Rouskas [35]. The multimessage multicasting problem based on the telephone communication mode has been studied under the name of *data migration with cloning* by Khuller et al. [36]. The total communication time for optimal schedules for the telephone communication mode is considerably larger to the one over the multicasting communication mode. A generalization of this message dissemination problem allowing for limited broadcasting or multicasting was considered by Khuller et al. [37]. These problems are used to model networks of workstations and grid computing. These data migration problems have also been studied under the sum of weighted completion time objective function by Gandhi et al. [38]. This version of the message dissemination problem is based on the telephone communication mode.

The edge coloring problem has also been studied under the "sums of colors" objective function [39]. It is interesting to note that even though the edge coloring and open shop problems are equivalent when the objective is to minimize the number of colors used and the makespan of the schedule, the same relationship does not hold when the objective is to minimize the "sum of the colors" (when colors are represented by integers) and the sum of completion time for the jobs. The correspondence is between the average of the colors and the average completion time for the tasks (which is different from the one for the jobs). The weighted versions of these problems have also been studied.

36.3 Multimessage Multicasting

We begin in Section 36.3.1 by presenting a reduction that shows that the decision version of the multimessage multicasting problem defined over complete networks is an NP-complete problem. Then we present in Section 36.3.2 upper and lower bounds for the TCT of the MM_C problem. Specifically we discuss a simple algorithm that shows that for every problem instance a communication schedule with TCT at most d^2 can be constructed in linear time. Then we show that there are problem instances for which all communication schedules have TCT at least d^2. Since these bounds are large we seek ways to improve on these results. In Section 36.3.3, we show that by adding buffers at the receiving end of every processor one may decrease the TCT required to solve MM_C problems. Specifically, we show that if there are l buffers at the receiving end of each processor it is possible to construct communication schedules with TCT at most $d^2/l + l - 1$. The algorithm corresponds to the one given in Section 36.3.2 when $l = 1$, and it is a generalization of that algorithm for $l > 1$. When $l = d$, the TCT reduces to $2l - 1$. The lower bounds in Section 36.3.2 apply when $l = 1$, but the instances involved in the lower bound have huge number of processors and messages have very large fan-out. In Section 36.3.4, we discuss algorithms for problems where the messages have restricted fan-out.

36.3.1 NP-Completeness of the *MM*$_C$

Gonzalez [1] established that the decision version of the multimessage multicasting problem is NP-complete even when every message is sent to at most two destinations. The reduction is from the edge

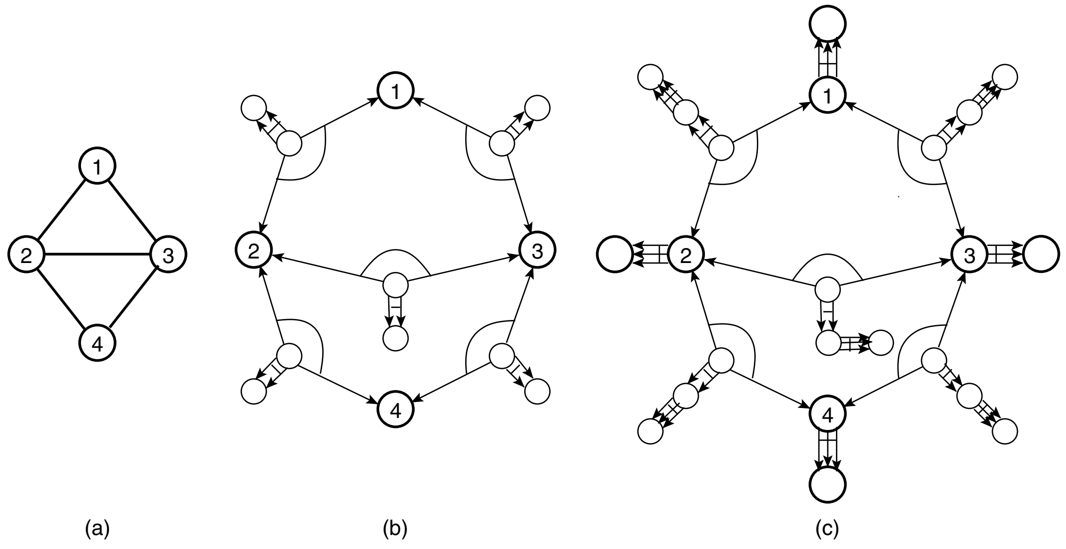

FIGURE 36.2 Graph edge coloring instance (a), corresponding MM_C instance (b), and corresponding instance of the MM_C problem with forwarding (c).

coloring problem. Given an undirected graph $G = (V, E)$ in which each vertex is of (graph) degree d, the edge coloring problem is to decide whether or not the edges of the graph can be colored with at most d colors so that every two edges incident to the same vertex are colored differently is an NP-complete problem (Holyer [40]).

The idea behind the reduction from the edge coloration problem to the MM_C is not too complex. Given an instance (G, d) of the edge coloration problem, one constructs an instance of the MM_C problem with fan-out $k = d$ as follows. Every vertex in G is represented by a processor, and for every edge there are two processors called *edge* and *extra-edge*. An edge in G is represented by d messages emanating from the edge processor representing the edge. One message has as destination the two processors representing the vertices in G that the edge joins. The other $d - 1$ messages have the same destination which is the corresponding extra-edge processor. Figure 36.2a gives a graph G with $d = 3$ and Figure 36.2b gives the instance of MM_C constructed from it. The claim is that the instance of the MM_C problem has a schedule with TCT equal to d iff the edges of graph G can be colored with exactly d colors. The equivalence between the problems can be established by observing that in a schedule with total communication d each of the messages emanating from an edge processor must be transmitted during a single communication round. Therefore, the message with two destinations must be multicasted to its two destinations during a single communication round. So it corresponds to the edge in G and thus both problems are equivalent. Clearly, the reduction takes polynomial time with respect to the number of vertices and edges in G. A formal proof for Theorem 36.1 is given in Reference 24.

Theorem 36.1 *The decision version of the MM_C problem is NP-complete even when $k = 2$.*

36.3.2 Upper and Lower Bounds for the MM_C

We show that every instance of the MM_C of degree d has a communication schedule with total communication time at most d^2. Furthermore, one such schedule can be constructed in linear time with respect to the total number of message destinations. Then we present a problem instance, which can be generalized to be of degree $d > 2$, such that all its communication schedules have TCT at least d^2.

Gonzalez algorithm [1] constructs a communication schedule with TCT at most d^2 for every instance of the MM_C problem of degree d. The algorithm begins by defining d^2 colors with a pair of integers i and

j with values between 1 and d. First all the messages emanating out of each processor are arranged in some order (any order) and the edges incident to every processor are also arranged in some order (any order is fine). We select one edge at a time. When considering edge $e = (p, q)$ it is colored with the color (i, j) if the edge e represents a message-destination pair corresponding to the i-th message originating at processor P_p and it is the j-th edge incident upon processor P_q. No two edges incident upon a vertex have the same second color component, and not two edges originating at the same processor that represent different messages are assigned the same first color component. Therefore the coloring just defined is valid. From this coloring one can easily construct a schedule for the MM_C problem with TCT equal to d^2. Furthermore, this schedule can be constructed in linear time with respect to the input length. These claims are summarized in the following theorem whose proof follows the arguments defined above and is given in Reference 1.

Theorem 36.2 *The above algorithm generates a communication schedule with TCT at most d^2 for every instance of the MM_C problem of degree d. The time complexity for the algorithm is linear with respect to the total number of message destinations.*

Now let us establish a lower bound for the TCT for problem instances of degree d. We show that there are problem instances such that all their schedules have TCT at least d^2. Consider the instance of the MM_C problem with $d = 2$ given in Figure 36.3. The problem instance consists of 28 processors. Each of the four rectangles represents a different processor and the solid black dots represent 24 additional processors. Each of the rectangle processors sends two messages which are labeled with the two letters. We claim that the problem instance does not have a schedule with TCT less than four. We prove this by contradiction. Suppose there is a schedule with total communication time three. Then each of the four rectangle processors must send one of its messages during exactly one communication round. But since there are four of those messages and only three communications rounds, we know that two of such messages must be transmitted during the same communication round. Any two of the messages originating at different processors have a destination in common. Therefore, there does not exist a communication schedule with TCT at most three. So all communication schedules have TCT at least $d^2 = 4$.

Gonzalez [1] defined instances of the MM_C problem for all $d > 2$ such that all their communication schedules have TCT at least d^2. The proof is a generalization for the one discussed above. The following theorem established in Reference 1 summarizes the discussion.

Theorem 36.3 *There are MM_C problem instances of degree d, for all $d > 1$, such that all their communication schedules have TCT at least d^2.*

The instance given above for $d = 2$ has 28 processors, but the one for $d = 4$ has more than one million processors. These problem instances have messages with a very large fan-out in order to achieve the d^2 bound. These are scenarios that are not likely to arise in commercial systems in the near future. There are different ways to construct schedules with smaller TCT. In Section 36.3.3, we discuss the case when there are buffers and in Section 36.3.4 we discuss algorithms for problem instances with fixed fan-out. This is a more likely scenario for the near future.

36.3.3 Receiving Buffers

One relatively inexpensive way to achieve high performance is by adding l buffers [41] at the receiving end of each processor and developing controlling hardware so the buffering behaves as follows: (i) if at least one buffer has a message during a communication round, then one such message (probably in a FIFO order) will be transferred to the processor and at the next communication round the buffer will be said to be free; and (ii) a processor may receive as many messages as the number of free buffers it has. If more messages are sent to a processor than the number of free buffers, then none of the buffers will receive the new messages and the processors that sent the message will be informed that their message did not

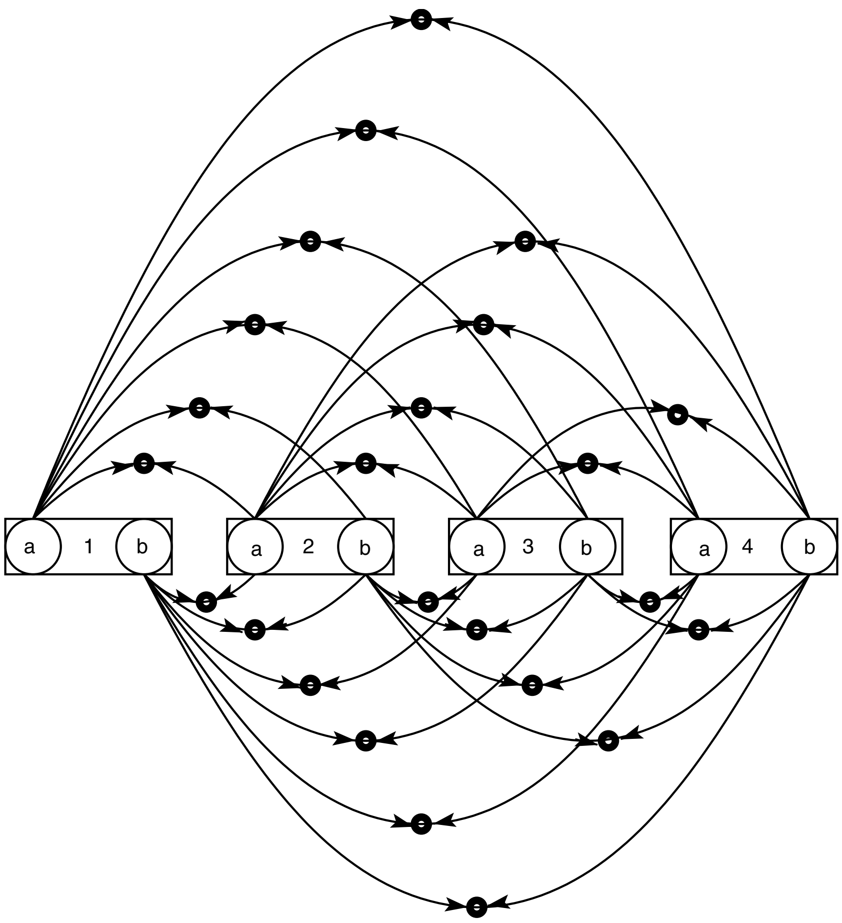

FIGURE 36.3 Problem instance of degree 2 that requires 4 colors. The triangles and solid circles represent processors.

reach all its destinations, but will not know the destinations that received the message. For this message dissemination problem, Gonzalez has recently developed [32] an efficient approximation algorithms that generates communication schedules with TCT at most $d^2/l + l - 1$. When there is one buffer, $l = 1$, the TCT is just d^2 and the algorithm reduces to the algorithm discussed in the previous subsection. For $l = d$, the TCT is $2d - 1$, which is slightly better than the one we will discuss in the next section where forwarding is allowed. The main advantage of this algorithm is that its time complexity is linear. The disadvantage is that the buffers and the hardware have an additional cost, but forwarding does not normally have any hidden costs. In what follows we outline the algorithm for the case when there are l buffers per processor. The algorithm, which is a generalization of the one discussed in the previous subsection, was initially reported in Reference 32.

Assume that d is a multiple of l (the number of buffers). The set $\{(i, j)|1 \leq i \leq d \text{ and } 1 \leq j \leq d/l\}$ defines a set of d^2/l colors. Number (with the integers $[1, d]$) in any order all the messages originating at each vertex and label all the edges for each message with the message number. The incoming edges to each processor are arranged in nondecreasing order of their labels. The first l incoming edges to a processor, with respect to the ordering just defined, are given the value of 1, the next l edges are given the value of 2, and so on. The color (i, j) is assigned to edge $e = \{p, q\}$, when e represents the i-th message originating at processor P_p, and e was assigned the value j as an edge incident to vertex P_q.

A communication schedule with TCT at most $d^2/l + l - 1$ is constructed from the above edge coloration as follows. The messages multicasted at time i are the ones represented by the edges colored $(i, 1)$, for

$1 \leq i \leq d$. Each processor has at most l incoming messages colored $(i, 1)$, so it follows that the processor buffers will not overflow. But it may be that the l messages for a processor fill up the buffers. The messages colored $(i, 2)$ will be sent at time $i + d$, for $1 \leq i \leq d$. Then next d time slots are for the messages colored $(i, 3)$, and so forth. The schedule overlaps the process of freeing the buffers for the messages colored (i, j) with the arrival of the messages $(i, j + 1)$, for all $1 \leq j < d/l$. All the $(i, 1)$ messages are received from the buffers by the processor by time $d + l - 1$, hence the last $l - 1$ time units overlap with the reception of the first $(i, 2)$ messages. We need to show that there will not be an overflow of the buffers on any of the processors. Let P_r be a processor that receives a messages colored $(1, 2)$. This message must be the first message originating at some processor. From the way we assigned colors to the incoming edges to processor P_r, we know that all the messages colored $(i, 1)$ that were received by P_r were colored $(1, 1)$. Therefore, the buffers are empty by the time the messages $(1, 2)$ are received. So even if l messages are received concurrently, there will be buffers to store them. Essentially the same argument can be used to show that there will be room in the buffers for all the messages colored $(i, 2)$. The proof that there are buffers available when the messages $(i, 3)$, $(i, 4)$, ..., $(i, d/l)$ arrive is similar. After all the messages are received one needs $l - 1$ time units to empty all the buffers. Therefore our procedure constructs a communication schedule with TCT at most $d * (d/l) + l - 1$. This algorithm and the proof of correctness appears in Reference 32, and it is summarized in the following theorem which we state without a proof. Readers interested in additional details are referred to Reference 32.

Theorem 36.4 *The above algorithm constructs a communication schedule with TCT at most $d^2/l + l - 1$ for every instance of the MM_C problem of degree d, where l is the number of buffers available at each processor. The time complexity for the algorithm is linear with respect to the total number of message destinations.*

For the case when $l = 1$, the lower bound constructed earlier in this chapter automatically provides the lower bound d^2. In other words, there are problem instances that require TCT of d^2, which is exactly the same as the upper bound provided by the schedules generated by the above algorithm. Gonzalez [32] has established the lower bound of $2d - 1$ for the case when there are $l = d$ buffers. This lower bound matches the upper bound for the TCT of the schedules generated by the above algorithm. However, for other values of l, the current lower and upper bounds do not match. We conjecture that the upper bounds are best possible. This remains as an interesting open problem.

36.3.4 Messages with a Fixed Number of Destinations

First we present approximation algorithms for the case when every message has at most two destinations, and then we discuss the case when every message is sent to at most k destinations.

36.3.4.1 Two-Destination Messages

As we established earlier, the MM_C problem where every message has at most two destinations is an NP-hard problem. In this subsection we present two approximation algorithms for this restricted version of the problem. The idea for the first algorithm is simple. We take every message with two destinations and break it into two messages, each with one destination. The resulting instance is an instance of the multimessage unicasting problem and of degree $2d$. As mentioned, before the multimessage unicasting problem, can be solved in polynomial time [1] and it has a schedule with TCT equal to $2d$. The time complexity to construct such schedule is $O(r(\min\{r, n^2\} + n \log n))$ time, where $r \leq dn$.

Another algorithm to construct a communication schedule with TCT equal to $2d - 1$ is presented in Reference 1. The algorithm is faster than the one just described. At each iteration the idea is to color all the messages originating at one of the processors. A maximal set of messages is selected such that each message can be colored with exactly one color. The remaining messages are colored with two colors. The two colors used by each message are obtained from a complete matching in a bipartite graph constructed from the current coloration. Hall's theorem is used to show that the bipartite graph has a complete matching. This result was initially established in Reference 1 and it is summarized in the following theorem.

Theorem 36.5 *Algorithm GM given in Reference 1 constructs for every instance of the MM_C problem with fan-out $k = 2$ and degree d a communication schedule with TCT at most $2d - 1$. The time complexity for the procedure is $O(nd^{2.5})$, where n is the number of processors.*

36.3.4.2 Multidestination Messages

As in the previous subsection, we can split every message with k destinations into k messages with one destination. Then by solving the MU_C problem we generate a schedule with TCT equal to kd. In what follows we present a set of algorithms all of which, except for the first one, generate schedules with TCT smaller than kd.

The following algorithms send every message during no more than q communication rounds, where q is an input value. The algorithms color the messages originating at P_1, then P_2, and so on. We describe first the algorithm for $q = 1$. When considering P_j each message destination pair originating at P_j has a potential conflict with at most $d - 1$ other messages with destination P_r. If all of these edges have been colored, there are at most $d - 1$ colors that cannot be used to color the current edge. The k edges for the message will have at most $k(d-1)$ colors, none of which can be used to color all the edges representing the message. These colors are called *destination-forbidden* colors. When every message is to be colored with exactly one color, then there are at most $d - 1$ *source-forbidden* colors. Therefore this coloring procedure may require at most $d + k(d - 1)$ different colors. This means that there is a schedule with TCT at most $d + k(d - 1)$. This bound is larger than the previous kd bound when $d > k$; however, the schedule can be constructed faster than the previous one.

To decrease the number of colors required by the above coloring strategy, one can increase the number of different colors used to color every message to two, that is, $q = 2$ which means the messages are split into at most two groups and each group is to be colored uniquely. This increases the number of source-forbidden colors and does not reduce the total number of destination-forbidden colors for each edge. But one may use any of the two colors for a message provided both of these colors are not destination-forbidden colors in any of the edges representing the message. For example, any message with two destinations cannot be colored when there are only $d - 1$ colors and we only take into consideration the destination-forbidden colors. The reason for this is that the overlap of destination-forbidden colors between the two edges representing the message can be $d - 1$, and therefore none of the $d - 1$ colors may be used to color the message. When there are d colors available it is possible to color the two edges for the message with one or two colors as follows. If one color is not a destination-forbidden color in both of the edges, then such color can be used to color the message. On the other hand when all the d colors are destination-forbidden colors in at least one of the edges, then at most $d - 2$ colors are destination-forbidden colors for both edges and the two remaining colors can be used to color the two edges for the message. What is the maximum number of colors, as a function to d and k, such that the k edges representing a message cannot be colored with any two colors when considering only destination-forbidden conflicts? As we just established, this value is $d - 1$ for $k = 2$. When $k = 3$ and $d = 9$ the maximum number is 12. The sets of destination-forbidden colors $\{1, 2, 3, 4, 5, 6, 7, 8\}$, $\{1, 2, 3, 4, 9, 10, 11, 12\}$, and $\{5, 6, 7, 8, 9, 10, 11, 12\}$ for each of the three edges can be used to show that no pair of the 12 colors can be used to color the three edges representing the message. The reason is that every pair of colors appears as a pair of destination-forbidden colors in at least one of the sets, so the pair of colors cannot be used to color the three edges. We claim that if there are 13 colors available it is possible to color the three edges using two colors. To prove this we show that there is always at least one color that is destination-forbidden in at most one of the three edges. This is because if the 13 colors are in the two lists, then the lists must contain at least 26 elements. But each of the three lists contain only eight elements. So there is at least one color that is destination-forbidden in at most one list. This color can therefore be used to color two of the edges, and the edge where it is destination-forbidden can be colored with another color since the list contains at most eight destination-forbidden colors. Therefore, it is possible to color all messages when 13 colors are available. Gonzalez [1] has established that the maximum number of colors such that no two of them can color completely a message is $d-1$ for $k = 2$, about $1.5(d - 1)$ for $k = 3$, and so forth. Gonzalez [1] also derived asymptotic bounds for this case $q = 2$, but for brevity we do not include this result.

The above analysis can be used for the following greedy algorithm initially proposed in Reference 1. Given any instance of the MM_C problem and a value of q, the algorithm colors the messages originating at P_1 then the ones originating at P_2, and so on. When coloring the messages originating at P_j it colors one message at a time. Each message is colored with at most q colors by using a greedy strategy. It selects first a color that can be assigned to the largest number of edges, then it selects a second color that can be assigned to the largest number of uncolored edges, and so on. Gonzalez [1] shows that all the messages can be colored, each with at most q colors, by using at most $qd + k^{1/q}(d-1)$ colors. The procedure takes $O(q \cdot d \cdot e)$ time, where $e \leq nd$ is total number of message destinations. A proof of this result is given in Reference 1 and it is summarized in the following theorem.

Theorem 36.6 *The above algorithm constructs in $O(q \cdot d \cdot e)$ time a communication schedule with TCT $qd + k^{1/q}(d-1)$ for the MM_C problem with fan-out k of degree d using at most q colors per message.*

Gonzalez [27] has developed improved approximation algorithms for the MM_C problem. These approximation algorithms are more complex and the proofs for their approximation bounds is complex because they involve the manipulation of lengthy inequalities. But the proof technique is similar to the ones in this subsection. For $q = 3$, the approximation ratio of the improved algorithm is about 10% smaller for $k = 15$, about 40% smaller for $k = 100$, than when $q = 2$.

36.4 Multimessage Multicasting with Forwarding

The MMF_C problem is the MM_C when *forwarding* is allowed. Forwarding means that messages may be sent indirectly through other processors even when there is a direct link from the source to the destination processor. Can forwarding allow significantly better solutions for our problem? Remember that in the MM_C problem the communication load is balanced, that is, every processor sends and receives at most d messages, and all possible direct links between processors exist. It seems then that forwarding will not generate significantly better solutions, that is, communication schedules with significantly smaller TCT. However, as we shall see in this section there is significant difference between the TCT of schedules for the same instance of the MM_C and MMF_C problems. It is important to point out that message forwarding consumes additional resources (links). The least amount of link capacity is utilized when forwarding is not permitted.

NP-completeness: The reduction in the previous section for the MM_C cannot be used to show that the decision version of the MMF_C problem is NP-complete. The reason is that the extra-edge processor may be used for forwarding in schedules with TCT d (see Figure 36.2b). This destroys the equivalence with the edge coloration problem. However, the reduction can be easily modified for the MMF_C problem. The idea is to introduce another processor, called *additional-edge*, for each edge in G (see Figure 36.2c). This additional-edge processor will receive d messages from the corresponding extra-edge processor. In any schedule with TCT equal to d the extra-edge processors will be busy all the time sending its own messages and will not be able to forward any of the messages originating at an edge processors. This observation can be used to establish the equivalence with the edge coloration problem. Readers interested in additional details are referred to References 1 and 8 for a formal proof for all cases, rather than just the informal arguments given above.

The most interesting aspect of the MMF_C problem is that for every problem instance one can construct in polynomial time a schedule with TCT at most $2d$ [8,26]. The two known algorithms to construct such schedules [8, 26] are slower than the ones we discussed for the MM_C problem in the previous section. However, the communication schedules have TCT that is significantly lower than that for the MM_C problem. The only exception is when every processor has $l = d$ buffers, but this requires additional hardware to store the data and control the buffer operations.

Both of the algorithms developed by Gonzalez [8, 26] have a two-phase structure. The first phase forwards messages in such a way that the resulting problem to be solved is a multimessage unicasting problem of degree d. As we have seen, this problem is an instance of the makespan openshop preemptive scheduling problem in which all tasks have unit execution time requirements. The second phase generates a communication schedule for this openshop problem from which a solution to the MMF_C can be obtained. The time complexity of the second phase is $O(r(\min\{r, n^2\} + n \log n))$, where r is the total number of message destinations ($r \leq dn$). The difference between the two algorithms is the first phase. The algorithm in Reference 26 tries to forward all the messages in the least number of communication rounds (no more than d), but the procedure is complex. The algorithm in Reference 8 is very simple, but performs all the message forwarding in d communication rounds. In what follows we discuss the simpler of the two algorithms through an example. Readers interested in additional details are referred to Reference 8.

The algorithm forwards all the messages. This is accomplished in d communication rounds. At each communication round every processor forwards at most one message and receives at most one message that it must deliver in the second phase to a set of at most d destinations. We explain this algorithm using the following example.

Example 36.2 *There are 7 processors and 15 messages. Table 36.4 lists the messages originating at each process and their destinations.*

The messages are ordered with respect to their originating processors. For Example 36.2 they are ordered in alphabetic order. Define message–destination pairs for all the messages and their destinations. Order the tuple first with respect to the ordering of the messages $A, B, \ldots, O$, and next with respect to their destinations (in increasing order). The messages in the first three pairs are forwarded from their originating processor to the first processor. The messages in the second three pairs are forwarded from their originating processor to the second processor, and so on. Clearly, each processor sends at most three messages, but the messages will be forwarded to one or more destinations. To avoid sending more than one message at a time to the same processors, the first three messages (A, B, C) must be multicast at time 1, 2, and 3, respectively; the next three messages (D, E, F) must be multicast at time 1, 2, and 3, respectively; and so on. Table 36.5 lists all the multicasting operations performed in phase 1 for Example 36.2. In our example, some messages are sent to multiple processors (G, M), while others are sent to single destinations. It is important to note that problem instances, like the one given above, some of the messages will end up being sent to and from the same processor. Obviously, these messages do not need to be transmitted. In some problem instances the elimination of these transmissions decreases the TCT, but in general it does not.

The resulting multimessage unicasting problem instance is given in Table 36.6. In Reference 8, a formal procedure is presented to perform the transformation outlined above. The algorithm is a generalization of

TABLE 36.4 Source Processor and Message–Destination Pairs for Example 36.2

Source	1	2	3	4	5	6	7
Message destination	—	A: {7}	C: {1}	E: {2, 3}	H: {1, 6}	J: {4}	M: {4, 5}
Message destination	—	B: {4}	D: {2}	F: {1, 7}	I: {6, 7}	K: {2}	N: {3}
Message destination	—	—	—	G: {5, 6}	—	L: {5}	O:{3}

TABLE 36.5 Message Forwarding Operations in Phase 1 for Example 36.2

Source	1	2	3	4	5	6	7
Round 1	—	$A \rightarrow \{1\}$	$D \rightarrow \{2\}$	$G \rightarrow \{3, 4\}$	—	$J \rightarrow \{5\}$	$M \rightarrow \{6, 7\}$
Round 2	—	$B \rightarrow \{1\}$	—	$E \rightarrow \{2\}$	$H \rightarrow \{4\}$	$K \rightarrow \{6\}$	$N \rightarrow \{7\}$
Round 3	—	—	$C \rightarrow \{1\}$	$F \rightarrow \{3\}$	$I \rightarrow \{5\}$	$L \rightarrow \{6\}$	$O \rightarrow \{7\}$

TABLE 36.6 Resulting Multimessage Unicasting Problem Constructed for Example 36.2

Source	1	2	3	4	5	6	7
Message destination	A: {7}	D: {2}	F: {1}	G: {6}	I: {6}	K: {2}	M: {5}
Message destination	B: {4}	E: {2}	F: {7}	H: {1}	I: {7}	L: {5}	N: {3}
Message destination	C: {1}	E: {3}	G: {5}	H: {6}	J: {4}	M: {4}	O:{3}

the above procedure and it is shown that it constructs an instance of the multimessage unicasting problem whose solution provides a solution to the original problem instance [8].

The above construction is such that all the multicasting operations have as destination adjacent processors. This property is important because every communication round can also be performed by a pr-dynamic network. In other words, the schedule for a fully connected network can be used directly by any pr-dynamic network. Gonzalez [8] established the following theorem on the basis of a generalization of the above arguments.

Theorem 36.7 *Given any instance of the MMF_C of degree d, the algorithm presented in Reference [8] constructs a communication schedule with TCT at most $2d$. The procedure takes $O(r(\min\{r, n^2\} + n \log n))$, where r is the total number of messages destinations ($r \leq dn$) and the communication schedule generated can be used for any pr-dynamic network.*

It is important to note that the algorithms in the previous section are faster than the one discussed in this section. However, the schedules generated by the algorithm in this section have considerably smaller TCT, and are also valid for pr-dynamic networks [8].

For restricted versions of the MMF_C it is possible to generate schedules with smaller TCT. Let us discuss briefly these versions of the problem. The $l - MMF_C$, for $2 \leq l \leq d$, is the MMF_C where every processor sends messages to at most ld destinations. Gonzalez [8] discusses an algorithm to construct communication schedule with TCT at most $\lfloor(2 - 1/l)d\rfloor + 1$ for the $l - MMF_C$ problem. For $l = 2$ the approximation ratio of the algorithm reduces from 2 to 1.5, but the difference becomes smaller as l increases. The algorithm is similar in nature to the one in this subsection. The lower TCT is a result of exploiting the limitation on the total number of message destinations.

A generalization of the MMF_C problem is called the *Multisource* MMF_C. The difference is that messages may originate in several processors, rather than just in one. Gonzalez [8] presents an algorithm for this problem. The algorithm constructs an instance of the MMF_C problem of least possible degree by selecting an origin for each message. This origin for each message can be determined by finding matchings in several bipartite graphs.

36.5 Distributed or On-Line Multimessage Multicasting

The $DMMF_C$ problem is the *distributed* or *online* version of the MMF_C problem. In this generalization of the MMF_C problem processors know local information and almost no global information. That is, each processor knows the destination of all the messages that it must send, and the values for d and n. Every processor in the MMF_C problem knows the messages it will receive, but in the $DMMF_C$ this information needs to be computed and transmitted to every processor. The schedule must be constructed online, and that is a time-consuming process. To speed-up the process, the above information and the schedule are not computed explicitly, but the operations performed by the algorithm form an implicit schedule.

The algorithm in Reference 28 for the $DMMF_C$ problem is based on the following idea. First it computes some global information and uses it to perform the first phase of the forwarding algorithm discussed in the previous section. Then it solves the resulting problem, which is the distributed multimessage unicasting problem, with forwarding, $DMUF_C$, problem.

The *DMUF$_C$* problem is a fundamental problem for optical-communication parallel computers [10,42–44]. Valiant's [44] distributed algorithm constructs a communication schedule with $O(d + \log n)$ expected TCT. The expected TCT is optimal, to within a constant, for $d = \Omega(\log n)$. For $d = o(\log n)$, Goldberg et al. [43] showed that no more than $O(d + \log\log n)$ communication rounds are needed with certain probability. When forwarding is not allowed, the expected total communication time is larger, $\Theta(d + \log n \log\log n)$.

Gonzalez's [28] approach for the *DMMF$_C$* problem begins by computing some global information through the well-known parallel prefix algorithm. Then the information is used to execute the forwarding phase of the algorithm discussed in the previous section. As in the case of the forwarding algorithm discussed in the previous section, the resulting problem is simply a multimessage unicasting problem. This problem can be solved by using Valiant's [44] distributed algorithm mentioned above. For every instance of the *DMMF$_C$* problem the distributed algorithm performs all the communications in $O(d + \log n)$ expected total communication time [28]. The following theorem was established in Reference 28 by using a generalization of the analysis discussed in this section.

Theorem 36.8 *[28] For every instance of the DMMF$_C$ problem the algorithm given in [28] constructs a schedule with $O(d + \log n)$ expected TCT.*

Clearly, the *DMMF$_C$* problem is much more general than the *MMF$_C$* problem, but communication schedules for the *DMMF$_C$* problem have expected total communication time $\Omega(d + \log n)$, but the TCT for the *MM$_C$* problem is d^2, and just $2d$ for the *MMF$_C$* problem. The knowledge ahead of time of all the communication patterns permits us to construct schedules with significantly smaller TCT. Forwarding is a very important factor in reducing the TCT. This is the most important consequence of the results discussed in this chapter. If the communication is restricted to the telephone mode, the best possible schedules will have TCT that is not bounded by any function in terms of d. In fact when there is only one message (to be sent from one processor to all other processors) the TCT is about $\log n$ and that is best possible. So the multicasting communication primitive delivers performance that is not attainable under the telephone communication mode. Even though the scheduling problems are NP-hard, schedules with suboptimal TCT can be constructed in polynomial time. In other words, the approximation problem is tractable.

36.6 Message Dissemination over Arbitrary Networks

Generating schedules with near optimal TCT is a complex problem for message dissemination problems over arbitrary graph. One problem is that the degree of the problem is no longer a good lower bound for the optimal TCT. Gonzalez [30,31] has had some success with restricted message dissemination problems. More specifically, with the gossiping problem. In what follows we discuss the gossiping problem when the multicasting communication primitive is available.

As we defined in the first section, the *broadcasting* problem defined over an n processor network N consists of sending a message from one processor in the network to all the remaining processors. The *gossiping* problem over a network N consists of broadcasting n messages each originating at a different processor. The network in this case is an arbitrary one, so messages need to be forwarded and the messages are routed through paths that are not necessarily the ones with the fewest number of links.

Gossiping problems can be solved optimally when the network has a Hamiltonian circuit, but finding a Hamiltonian circuit in an arbitrary network is a computationally intractable problem. Fortunately, it is not necessary for a network to have a Hamiltonian circuit for the gossiping problem to be solvable in $n-1$ steps, where n is the number of nodes in the network. There are networks that do not have a Hamiltonian circuit in which gossiping can be performed in $n - 1$ communication steps under the multicasting communication mode, but not under the unicasting mode [30, 31]. This shows again the power of the multicasting communication mode. Not all networks allow for communication schedules with total communication time $n - 1$. For example, the straight line network does not have one such schedule because it is impossible to deliver a new message to each end of the line of processors at each communication round.

The broadcasting and gossiping problems have been studied for the past three decades [45]. However, most of the work is for the telephone type of communication. Most of the previous algorithms are not scalable because the communication mode allows for the transmission of packets of size 1 and n in one communication round. Under the traditional communication modes these problems are computationally difficult, that is, NP-hard. However, there are efficient algorithms to construct optimal communication schedules for restricted networks under some given communication modes [46–48]. There is no known polynomial time approximation algorithm with an approximation ratio bounded by a constant for the broadcasting problem defined over arbitrary graphs. The best approximation algorithms appear in References 48 and 49. A randomized algorithm is proposed in Reference 50. Broadcasting under the multicasting communication mode is a trivial problem to solve. That is, it is possible to construct a schedule with optimal TCT in linear time, for every problem instance.

A variation of the gossiping problem in which there are costs associated with the edges and there is a bound on the maximum number of packets that can be transmitted through a link at each time unit has been studied in Reference 51. Approximation algorithms for several versions of this problem with nonconstant approximation ratio are given in References 51–53.

We discuss now the algorithms for gossiping under the multicasting mode given in References 30 and 31. These algorithms take polynomial time. The algorithms begin by constructing a spanning tree of minimum radius, a problem that can be solved in polynomial time [30,31]. Then all the communications are performed in the resulting tree network. A vertex whose closest leaf is the farthest is called the *root* of the tree. The number of nodes in the network is n, and r is the radius of the network.

The simplest of algorithms [30], which has been used to solve other message routing problems, performs the gossiping in $2n + r$ time units. The idea is to send to the root all the messages and then send them down using the multicasting operation at each node. The best of the algorithms [31] is based on the observation that all the other operations can be carried out in a single stage by maximizing the concurrent operations performed at each step and finding appropriate times to transmit all messages down the tree. The total communication time of the schedules generated by this algorithm is just $n + r$. A lower bound of $n + r - 1$ for the number of communication rounds has been established for a large number of problem instances. Therefore, the algorithm given in Reference 31 generates schedules with TCT that is as close to optimal as possible for this set of problem instances. The *weighted gossiping* problem is a generalization of the gossiping problem where each processor needs to broadcast a nonempty set of messages, rather than just one. The currently best algorithm for the unrestricted version of the problem can be easily adapted to solve the weighted gossiping problem [31]. Furthermore, the algorithm can be easily transformed into a distributed algorithm. An interesting open problem is to design an efficient algorithm for the case when some of the weights could be zero.

36.7 Multimessage Multicasting in Optical Networks

Let us now consider the multimessage multicasting under the multiport communication model that allows multicasting. The architecture studied in this case is the star network with multiple fibers between nodes that allows optical switching between fibers along the same wavelength. The specific problem we consider is given any $(n + 1)$-node star network, a predetermined number of fibers that connect its nodes, and a set of multicasts (or multidestination messages) to be delivered in one communication round, find a conflict free message transmission schedule that uses the least number of wavelengths per fiber [33]. When the number of wavelengths, r, needed exceeds the number available, $\lambda_{\min}$, one may transform the schedule into one with $\lceil r/\lambda_{\min} \rceil$ communication phases or rounds over the same network, but restricted to $\lambda_{\min}$ wavelengths per fiber.

High transmission speeds are needed for many applications [54]. It is very likely that future communication networks will include a large amount of multidestination traffic [54,55]. Furthermore, since one of the biggest costs when building an optical network is the actual physical laying of the optical fibers, often many fibers may be installed at the same time, for about the same cost, resulting in multifiber networks [56]. Wide area testbeds are currently being developed, to transmit data over numerous wavelengths in realtime [57].

For the multimessage unicasting problem over a g fiber star network Li and Simha [58] developed a polynomial time algorithm to route all messages using $\lceil d/g \rceil$ wavelengths per fiber. Clearly, multifibers allow for solutions using fewer wavelengths, but the complexity of the switches needed for switching between fibers increases. Upper and lower bounds have been developed for the number of wavelengths needed to transmit all messages in one communication round (which translates directly to minimizing the number of communication rounds when the number of wavelengths per fiber is a fixed constant) [33]. Brandt and Gonzalez [33] have established an upper bound for m, the number of wavelengths per fiber, equal to $\lceil \frac{d^2}{i(g-1)} \rceil$, where i is any positive integer such that $1 \leq i \leq g-1$. The idea is to partition the fibers incoming into every processor into two groups: *receiving* and *sending* fibers. As their name indicates, the i receiving fibers are used to receive messages and the $g - i$ sending fibers are used for sending multicasts and the switching is at the center node. The optical switching can be set to deliver the messages from the sending fiber to the receiving fiber provided that they are sent and received along the same wavelength. To guarantee that this is always possible, we assign multiple wavelengths for the reception of each message on one of the receiving fibers, and we send along multiple wavelengths each message on one of the fibers. Specifically, the set of wavelengths is partitioned into d/i sets each with $m/(d/i)$ wavelengths. Each of these sets is called a *receiving set* of wavelengths. Now each multicast is sent on one wavelength from each of these sets. This guarantees that a switching to send the multicast to all its destinations always exists. The product $m \cdot g$ is minimum when $i = g/2$. In this case $m = \lceil 4d^2/g^2 \rceil$, which is about four times the lower bound $m = \lceil \frac{d^2}{g^2} + \frac{d}{g} \rceil$ we have established in Reference 33. For restricted values of g and d, Brandt and Gonzalez [33] have proved tight bounds for m, some of which are quite elaborate to establish. The reader is referred to Reference 33 for further information.

36.8 Discussion

It is simple to see that the *DMMF*$_C$ problem is more general than the *MMF*$_C$ and the *MM*$_C$ problems, but the best communication schedule for the *DMMF*$_C$ problem has TCT $\Omega(d + \log n)$ whereas for the *MMF*$_C$ problem is just $2d$, and the *MM*$_C$ problem is d^2. Therefore, knowing all the communication information ahead of time allows one to construct significantly better communication schedules. Also, forwarding plays a very important role in reducing the total communication time for our scheduling problems.

The most important open problem is to develop distributed algorithms with similar performance guarantees for processors connected through pr-dynamic networks. Algorithms exist for the nondistributed version of this problem [8]. The main difficulty in extending that work to the distributed case is the construction of the routing tables with only local information.

Another very challenging open problem is to develop efficient approximation algorithms for the *MMF*$_C$ problem that generate schedules with communication time significantly smaller than $2d$. There are several variations of the *MM*$_C$ problem that are worth studying. For example the case when there are precedence constraints between the messages seems to be one that arises in several applications.

References

[1] Gonzalez, T.F., Complexity and approximations for multimessage multicasting, *Journal of Parallel and Distributed Computing*, 55, 215, 1998.

[2] Bertsekas, D.P., and Tsitsiklis, J.N., *Parallel and Distributed Computation: Numerical Methods*, Prentice Hall, Englewood Cliffs, NJ, 1989.

[3] Krumme, D. W., Venkataraman, K.N., and Cybenko, G., Gossiping in minimal time, *SIAM Journal on Computing* 21(2), 111, 1992.

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

[5] Lee, T.T., Non-blocking copy networks for multicast packet switching, *IEEE Journal of Selected Areas of Communication*, 6(9), 1455, 1988.

[6] Liew, S.C., A general packet replication scheme for multicasting in interconnection networks, *Proceedings of the IEEE INFOCOM*, 1, 1995, 394.

[7] Turner, J.S., A practical version of Lee's multicast switch architecture, *IEEE Transactions on Communications*, 41(8), 1993, 1166.

[8] Gonzalez, T.F., Simple multimessage multicasting approximation algorithms with forwarding, *Algorithmica*, 29, 511, 2001.

[9] Gonzalez, T.F. and Sahni, S., Open shop scheduling to minimize finish time, *JACM*, 23(4), 665, 1976.

[10] Anderson, R.J. and G. L. Miller, G.L., Optical communications for pointer based algorithms, TRCS CRI 88 – 14, USC, 1988.

[11] Cole, R. and Hopcroft, J., On edge coloring bipartite graphs, *SIAM Journal on Computing*, 11(3), 540, 1982.

[12] Gabow, H. and Kariv, O., Algorithms for edge coloring bipartite graphs and multigraphs, *SIAM Journal Computing*, 11, 117, 1982.

[13] Cole, R., Ost, K., and Schirra, S., Edge-coloring bipartite multigraphs in $O(ElogD)$, *Combinatorica*, 21, 5, 2001.

[14] Gonzalez, T.F., Open shop scheduling, in *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, Leung, Y.J.-T., (ed) Chapman & Hall, CRC, 2004, Chap. 6.

[15] Coffman, Jr, E.G., Garey, M.R., Johnson, D.S., and LaPaugh, A.S., Scheduling file transfers in distributed networks, *SIAM Journal on Computing*, 14(3), 744, 1985.

[16] Whitehead, J., The complexity of file transfer scheduling with forwarding, *SIAM Journal on Computing* 19(2), 222, 1990.

[17] Choi, H.-A. and Hakimi, S.L., Data transfers in networks with transceivers, *Networks*, 17, 393, 1987.

[18] Choi, H.-A. and Hakimi, S.L., Data transfers in networks, *Algorithmica*, 3, 223, 1988.

[19] Hajek, B., and Sasaki, G., Link scheduling in polynomial time, *IEEE Transactions on Information Theory*, 34(5), 910, 1988.

[20] Gopal, I.S., Bongiovanni, G., Bonuccelli, M.A., Tang, D.T., and Wong, C.K., An optimal switching algorithm for multibeam satellite systems with variable bandwidth beams, *IEEE Transactions on Communications*, 30(11), 2475, 1982.

[21] Rivera-Vega, P.I., Varadarajan, R., and Navathe, S.B., Scheduling file transfers in fully connected networks, *Networks*, 22, 563, 1992.

[22] Goldman, A., Peters, J. G., and Trystram, D., Exchanging messages of different sizes, *Journal of Parallel and Distributed Computing*, 66, 18, 2006.

[23] Hall, J., Hartline, J., Karlin, A.R., Saia, J., and Wilkes, J., *Proc. of SODA*, 2001, 620.

[24] Gonzalez, T. F., MultiMessage Multicasting, *Proc. of the Third International Workshop on Parallel Algorithms for Irregularly Structured Problems,* LNCS 1117, Springer, 1996, 217.

[25] Gonzalez, T.F., Proofs for improved approximation algorithms for multimessage multicasting, UCSB TRCS-96-17, 1996.

[26] Gonzalez, T.F., Algorithms for multimessage multicasting with forwarding, *Proc. of the* 10*th PDCS,* 1997, 372.

[27] Gonzalez, T.F., Improved approximation algorithms for multimessage multicasting, *Nordic Journal on Computing,* 5, 196, 1998.

[28] Gonzalez, T.F., Distributed multimessage multicasting, *Journal of Interconnection Networks*, 1(4), 303, 2000.

[29] Gonzalez, T. F., On solving multimessage multicasting problems, *International Journal of Foundations of Computer Science*, Special Issue on Scheduling—Theory and Applications, 12(6), 791, 2001.

[30] Gonzalez, T. F., Gossiping in the multicasting communication environment, *Proc. of IPDPS*, 2001.

[31] Gonzalez, T. F., An efficient algorithm for gossiping in the multicasting communication environment, *IEEE Transactions on Parallel and Distributed Systems,* 14(7), 701, 2003.

[32] Gonzalez, T. F., Improving the computation and communication time with buffers, *Proc. 17th IASTED PDCS,* 2006, 336.

[33] Brandt, R. and Gonzalez, T.F., Multicasting using WDM in multifiber optical star networks, *Journal of Interconnection Networks*, 6(4), 383, 2005.

[34] Shen, H., Efficient multiple multicasting in hypercubes, *Journal of Systems Architecture,* 43(9), 1997.

[35] Thaker, D. and Rouskas, G., Multi-destination communication in broadcast WDM networks: A survey, *Optical Networks*, 3(1), 34, 2002.

[36] Khuller, S., Kim, Y.-A., and Wan, Y-C., Algorithms for data migration with cloning, *SIAM Journal on Computing*, 33(2), 448, 2004

[37] Khuller, S., Kim, Y.-A., and Wan, Y-C., Broadcasting on networks of workstations, *Proc. of SPAA*, 2005.

[38] Gandhi, R., Halldorsson, M.M., Kortsarz, M., and Shachnai, H., Improved results for data migration and open shop scheduling, *ACM Transactions on Algorithms,* 2(1), 116, 2006.

[39] Bar-Noy, A., Bellare, M., Halldorsson, M.M., Shachnai, H., and Tamir, T., On chromatic sums and distributed resource allocation, *Inf. Comput.*, 140, 183, 1998.

[40] Holyer, I., The NP-completeness of edge-coloring, *SIAM Journal of Computing,* 11, 117, 1982.

[41] Bruno, J., Personal communication, 1995.

[42] Gereb-Graus, M. and Tsantilas, T., Efficient optical communication in parallel computers, *Proc. of 4th SPAA,* 1992, 41.

[43] Goldberg, L.A., Jerrum, M., Leighton, T., and Rao., S., Doubly logarithmic communication algorithms for optical-communication parallel computers, *SIAM J. Comp.*, 26(4), 1100, 1997.

[44] Valiant, L.G., General purpose parallel architectures, in *Handbook of Theoretical Computer Science,* van Leeuwen, J., ed., Elsevier, 1990, Chap. 18.

[45] Hedetniemi, S., Hedetniemi, S., and Liestman, A survey of gossiping and broadcasting in communication networks, *NETWORKS,* 18, 129, 1988.

[46] Even, S. and Monien, B., On the number of rounds necessary to disseminate information, *Proc. SPAA,* 1989, 318.

[47] Fujita, S. and Yamashita, M., Optimal group gossiping in hypercube under circuit switching model, *SIAM Journal on Computing* 25(5), 1045, 1996.

[48] Ravi, R., Rapid rumor ramification, *Proc. FOCS,* 1994, 202.

[49] Hromkovic, J., Klasing, R., Monien, B., and Peine, R., *Dissemination of Information in Interconnection Networks (Broadcasting and Gossiping)*, Du D.Z. and Hsu, D.F., eds., Kluwer Academic, 1995, 273.

[50] Feige, U., Peleg, D., Raghavan, P., and Upfal, E., Randomized broadcast in networks, *Proc. of SIGAL,* LNCS, Springer-Verlag, 1990, 128.

[51] Fraigniaud, P. and Vial, S., Approximation algorithms for broadcasting and gossiping, *Journal of Parallel and Distributed Computing,* 43, 47, 1997.

[52] Bermond, J. C., Gargano, L., Rescigno, C.C., and Vaccaro, U., Fast gossiping by short messages, *SIAM Journal on Computing* 27(4), 917, 1998.

[53] Gargano, L., Rescigno, A.A., and Vaccaro, U., Communication complexity of gossiping by packets, *Journal of Parallel and Distributed Computing,* 45, 73, 1997.

[54] Rouskas, G. and Ammar, M., Multi-destination communication over single-hop lightwave WDM networks, *Proc. of INFOCOM*, 1994, 1520.

[55] Rouskas, G. and Ammar, M., Multi-destination communication over tunable-receiver single-hop WDM networks, *IEEE Journal on Selected Areas in Communications*, 15(3), 501, 1997.

[56] Ferreira, A., Perennes, S., Richa, A., Rivano, H., and Stier, N., On the design of multifiber WDM networks, *Proc. of AlgoTel*, 2002, 25.

[57] Chien, A., OptIPuter software: System software for high bandwidth network applications. http://www-csag.ucsd.edu/projects/Optiputer.html, 2002.

[58] Li, G. and Simha, R., On the wavelength assignment problem in multifiber WDM star and ring networks, *IEEE/ACM Transactions on Networking*, 9(1), 60, 2001.