COMPLEXITY RESULTS FOR COLLECTIVE COMMUNICATIONS ON HETEROGENEOUS PLATFORMS

O. Beaumont[1]
L. Marchal[2]
Y. Robert[2]

Abstract

In this paper, we consider the communications involved in the execution of a complex application, deployed on a heterogeneous platform. Such applications extensively use macro-communication schemes, for example to broadcast data items, either to all resources (broadcast) or to a restricted set of targets (multicast). Rather than aiming at minimizing the execution time of a single collective communication, we focus on the steady-state operation. We assume that there is a large number of messages to be broadcast or multicast in pipelined fashion, and we aim at maximizing the throughput, i.e. the (rational) number of messages which can be broadcast or multicast every time-step. We target heterogeneous platforms, modeled by a graph where resources have different communication and computation speeds. Achieving the best throughput may well require that the target platform is used in totality: different messages may need to be transferred along different paths.

The main focus of the paper is on complexity results. We aim at presenting a unified framework for analyzing the complexity of collective communication schemes. We concentrate on the classification (whether maximizing the throughput is a polynomial or NP-hard problem), rather than actually providing efficient polynomial algorithms (when such algorithms are known, we refer to bibliographical pointers).

Key words: scheduling, collective communications, NP-completeness, broadcast, heuristics, heterogeneous clusters, grids

1 Introduction

Collective communication schemes in computer networks are the focus of a vast literature. The one-to-all broadcast, or single-node broadcast (Kumar et al. 1994), is the most primary collective communication pattern. Initially, only the source processor has the data that need to be broadcast; at the end, there is a copy of the original data residing at each processor. Parallel algorithms often require the sending of identical data to all other processors, in order to disseminate global information (typically, input data such as the problem size or application parameters). Numerous broadcast algorithms have been designed for parallel machines such as meshes, hypercubes, and variants (see, among others, Johnsson and Ho 1989; Watts and Van De Geijn 1995; Tseng, Wang, and Ho 1999; Ko, Latifi, and Srimani 2000; Wang and Tseng 2000). The one-to-all message passing interface (MPI) routine (Snir et al. 1996) is widely used, and particular attention has been paid to its efficient implementation on a large variety of platforms (Hwang and Xu 1998).

Multicasting also is a key communication primitive in computer networks (Kumar and Jaffe 1983). Lin and Ni (1993) have published a survey paper where they consider different multicast algorithms operating under several network models. They explain the close relationships between multicast algorithms and Steiner trees (see Winter 1987 for an overview of Steiner problems). Several authors have discussed optimized broadcast algorithms for a variety of parallel architectures, such as wormhole routed networks (Robinson, McKinley, and Cheng 1995), cut-through routed networks (Cohen et al. 1998), and networks of workstations (Sivaram et al. 2001). Recently, the design of multicast algorithms has been the focus of many papers, due to the advent of new technologies such as mobile (Anastasi, Bartoli, and Spadoni 2001), wireless (Wieselthier, Nguyen, and Ephremides 2002), ad hoc (Gopalsamy et al. 2002), and optical (Yang, Wang, and Qiao 2000) networks.

There are three main variants considered in the literature.

Atomic: the source message is atomic, i.e. cannot be split into packets. A single message is sent by the source processor, and forwarded across the platform to target nodes.

Pipelined: the source message can be split into an arbitrary number of packets, which may be routed in a pipelined fashion, possibly using different paths.

The International Journal of High Performance Computing Applications, Volume 20, No. 1, Spring 2006, pp. 5–17
DOI: 10.1177/1094342006061877

[1]LABRI, UMR CNRS 5800 BORDEAUX, FRANCE

[2]LIP, UMR CNRS-INRIA 5668 ENS LYON, FRANCE
YVES.ROBERT@ENS-LYON.FR

Series: the same source processor sends a series of atomic messages, involving messages of the same size. The processing of these communication schemes can be pipelined.

For the first two problems, the goal is to minimize the total execution time (or makespan) of the schedule. For the third problem, the objective function is rather to optimize the throughput of the steady-state operation, i.e. the average amount of data broadcast or multicast per time-unit. In this paper we concentrate on the series problems, in particular the series of broadcasts and series of multicasts problems. Note that solving the series problem provides a solution to the corresponding pipelined problem. If we choose a suitable size for the packets, split the original message into packets, and schedule the series of packets, we can derive an asymptotically optimal algorithm. See Beaumont et al. (2004a) for an illustration of the pipelined broadcast problem. We focus on two different realistic models of communications:

- the bidirectional one-port model, where at any given time-step a given processor can simultaneously receive data from one of its neighbors, and send (independent) data to one of its neighbors;
- the unidirectional one-port model, where at any given time-step a given processor can either receive data from one of its neighbors or send (independent) data to one of its neighbors.

These models are to be contrasted with the traditional multiport model, where the number of simultaneous send or receive operations executed by a given processor is not bounded.

The series of broadcasts problems has been considered by Moore and Quinn (1997) and Desprez et al. (1993), but with a different perspective. They consider that distinct processor sources successively broadcast one message, and their goal is to load-balance this series of communications. Here, we assume that the same source processor initiates all the broadcasts. This is closer to a master–slave paradigm where the master disseminates the information to the slaves in a pipelined fashion; for instance, the data needed to solve a collection of (independent) problem instances.

The rest of the paper is organized as follows. Section 2 is devoted to the description of the architectural framework for the operation of the nodes, i.e. the unidirectional one-port and bidirectional one-port models. Section 3 is devoted to the formal specification of the series of broadcasts and series of multicasts problems. In Section 3.2.2 we prove the NP-completeness of the series of multicasts problem. Section 4 is devoted to a brief description of the ellipsoid method for solving linear programs, which will be used to derive polynomial algorithms for the series of broadcasts problems under both models. We give some extensions of the complexity results in Section 5. Finally, we state some concluding remarks in Section 6.

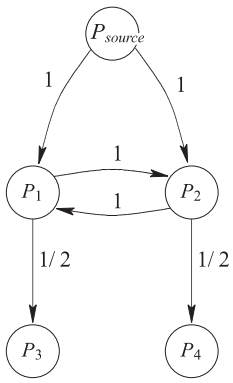

Fig. 1 Simple platform topology. The value of $c_{j,k}$ is indicated along each edge. The node P_{source} is the source of the broadcasts.

2 Framework

2.1 Network Model

The target architectural platform is represented by an edge-weighted directed graph $G = (P, E, c)$, as illustrated in Figure 1. Note that this graph may well include cycles and multiple paths. Let $p = |P|$ be the number of nodes. There is a source node P_{source}, which plays a particular role; it initially holds all the data to be broadcast or multicast. For the broadcast operation, all the other nodes P_i, $1 \leq i \leq p$, $i \neq s$, are destination (or target) nodes which must receive all the data sent by P_{source}. For the multicast operation, we denote by $\mathcal{D}$ the set of destination nodes which must receive all the data sent by P_{source}. Note that non-destination nodes may be used to transfer some messages to the set of target nodes.

Each edge $e_{j,k} : P_j \rightarrow P_k$ is labeled by a value $c_{j,k}$ which represents the time needed to communicate a unit-size message from P_j to P_k. The graph is directed, and the time to communicate in the reverse direction, from P_k to P_j, provided that this link exists, is $c_{k,j}$. Note that if there is no communication link between P_j and P_k we let $c_{j,k} = +\infty$, so that $c_{j,k} < +\infty$ means that P_j and P_k are neighbors in the communication graph.

There are several scenarios for the operation mode of the processors. In this paper, we concentrate on the unidirectional one-port model and the bidirectional one-port model.

2.2 Unidirectional One-Port Model

Under the assumption of the unidirectional one-port model, a processor can, at a given time-step, perform a single communication operation: it can either send data to one of its neighbor nodes or receive data from it. In this section, we state this communication model more precisely. If P_j sends a message of size L to P_k at time-step t, then:

- P_k cannot initiate another receive or send operation before time-step $t + c_{j,k} \times L$;
- P_j cannot initiate another send or receive operation before time-step $t + c_{j,k} \times L$.

Proposition 1. *A set of communications that can be handled in parallel in G under the one-port unidirectional model is associated with a matching in G.*

Proof. Consider a matching in G. Each node P_i has at most one neighbor node in this matching so that it can communicate without conflicts with this neighbor node. Conversely, consider a set of simultaneous communications in G under the one-port model. Each node P_i can communicate at the same time with at most one neighbor, so that the set of communicating nodes (P_i, P_j) defines a matching in G. □

In what follows, we denote by $\mathcal{M}_1^{(G)}$, …, $\mathcal{M}_{N_{\mathcal{M}(G)}}^{(G)}$ the (finite) set of matchings of G.

2.3 Bidirectional One-Port Model

In the bidirectional one-port model, we assume that a processor is able to perform at the same time a send operation to one of its neighbors, and an independent receive operation from another neighbor. To state this more precisely, if P_j sends a message of size L to P_k at time-step t, then:

- P_k cannot initiate another receive operation before time-step $t + c_{j,k} \times L$ but it can perform a send operation to a third node;
- P_j cannot initiate another send operation before time-step $t + c_{j,k} \times L$ but it can perform a receive operation from a third node.

Proposition 2. *A set of communications that can be handled in parallel in G under the one-port bidirectional model is associated with a matching in $G^{(B)}$, where $G^{(B)}$ is the graph depicted in Figure 2.*

Proof. There is an edge between P_j^{out} and P_k^{in} in $G^{(B)}$ if and only if $c_{j,k} < +\infty$. Consider a matching in $G^{(B)}$. We associate node P_i^{out} (P_i^{in}) with outgoing (incoming) communications of P_i. Each node P_i^{out} has at most one neighbor node in this matching so that P_i can perform an outgoing communication with this neighbor without conflict. Similarly, each node P_i^{in} has at most one neighbor node in this matching so that P_i can perform an incoming communication with this neighbor without conflict.

Conversely, consider a set of simultaneous communications in G under the bidirectional one-port model. At a given time-step, each node P_i sends a message to at most one neighbor (so that P_i^{out} has at most one neighbor in $G^{(B)}$) and it receives at most one message from at most one of its neighbor nodes (so that P_i^{in} has at most one neighbor in $G^{(B)}$). Therefore, a set of simultaneous communications in G under the one-port bidirectional model defines a matching in $G^{(B)}$. □

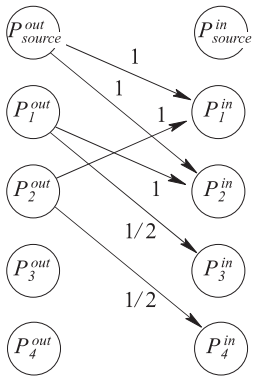

Fig. 2 Bipartite graph $G^{(B)}$ built from the platform graph G of Figure 1.

In what follows, we denote by $\mathcal{M}_1^{(B)}$, …, $\mathcal{M}_{N_{\mathcal{M}(B)}}^{(B)}$ the (finite) set of matchings of $G^{(B)}$.

3 Broadcast and Multicast

In this section, we first concentrate on two particular problems: the series of broadcasts and series of multicasts problems. In both problems, the same source processor sends a series of N atomic, same-size messages. The processing of these communication schemes can be pipelined. More precisely, we concentrate on throughput maximization. Our aim is to find a set of broadcast trees (to transfer the different parts of the message) and a set of matchings (to orchestrate the communications), so as to maximize the throughput, i.e. the fractional number of communications that can be initiated, at steady state, during one time-unit.

In order to find the actual schedule for series of broadcasts or multicasts, transient states (initialization and clean-

up phases) must be taken into account. The actual reconstruction of a valid schedule (including transient states) is based on potential graphs, and has been considered in Beaumont et al. (2004b) in a more general framework (schedule of a large number of independent task graphs). The proof of the correctness of a valid schedule consists in proving that the length of both transient phases can be bounded by B, where B does not depend on the overall number of tasks but on the platform graph only. Due to lack of space, the proof cannot be included in the paper, but we refer the interested reader to Beaumont et al. (2004b).

3.1 Series of Broadcasts

3.1.1 Formal definition In the case of the series of broadcasts problem, messages are sent to all the nodes of the platform. We are interested in maximizing the throughput, i.e. the number of messages that can be sent during each time-unit.

Consider a given atomic message among the series of messages to be broadcast. Clearly, there is no reason why a given node would receive the message twice. Thus, we can easily transform any solution where a message is received several times by a node into a solution where each node receives the message only once: we remove all the communications of the same atomic message received by each node but one. The resulting communication scheme involves a strict subset of the communications induced by the initial broadcast scheme, and therefore achieves a throughput equal to, or better than, the original throughput. Therefore, the broadcast of each atomic message can be implemented using a single spanning tree of the whole platform.

In the series of broadcasts problem, each atomic message in the series is transmitted along a broadcast tree. Different messages may be broadcast along different trees. However, note that the same tree may well be used to broadcast several atomic messages in the series. For example, only two trees $\mathcal{T}_1$ and $\mathcal{T}_2$ may be used, the odd indexed messages being broadcast along $\mathcal{T}_1$, while the even indexed messages are broadcast along $\mathcal{T}_2$.

We are looking for a description of the schedules based upon broadcast trees. Our aim is to find the set of weighted spanning trees of G that will be used to broadcast the set of atomic messages, where the weight of a tree is the number of messages in the series that are broadcast along the same tree.

Let us denote by $\mathcal{T}_1, \ldots, \mathcal{T}_T$ the set of all spanning trees of G rooted at P_{source}. T, the number of spanning trees, may be quite large but is nevertheless finite. We denote by N the number of messages in the series to be broadcast. Therefore, we are looking for a set of weighted spanning trees $\{(\alpha_t(N), \mathcal{T}_t^{broad}),\ t = 1 \ldots T\}$, where $\alpha_t(N)$ denotes the number of atomic messages (among the N atomic messages) that are broadcast along the spanning tree $\mathcal{T}_t^{broad}$. Of course, we need to ensure that $\sum_{t=1}^{T} \alpha_t(N) = N$.

The set of broadcast trees describes all the communications that have to take place in the schedule. However, these communications have to be orchestrated so that they comply with the underlying communication model. For example, if two broadcast trees share an edge, communications along this edge must be sequentialized. Also, one-port constraints, either unidirectional or bidirectional, have to be enforced in the schedule.

3.1.2 Bidirectional one-port model Using the bidirectional one-port model, we have seen that the set of communications that can be handled simultaneously corresponds to a matching in the bipartite graph $G^{(B)}$. To orchestrate all the communications, we are looking for a weighted set of matchings in this graph, denoted by $\{(\beta_m(N), \mathcal{M}_m^{(B)}),\ m = 1 \ldots M\}$. Each matching in this set represents a time interval of length $\beta_m(N)$ during which the edges of the matching will be used to communicate data.

Consider an edge (P_j, P_k) in G and the corresponding edge in the bipartite graph (P_j^{out}, P_k^{in}). This edge may well belong to several matchings. The total occupation time of this edge according to the weighted set of matchings described above is

$$\sum_{m,\ \mathcal{M}_m^{(B)} \ni (P_j^{out}, P_k^{in})} \beta_m.$$

This edge may also be used by several broadcast trees in the weighted set $\{(\alpha_t(N), \mathcal{T}_i^{broad}),\ t = 1 \ldots T\}$ described before, where $\alpha_t(N)$ denotes the number of atomic messages broadcast along $\mathcal{T}_t^{broad}$. Since the transfer of a unit size message along this edge lasts $c_{j,k}$ time-units, the total occupation time of this edge by this set of broadcast trees is

$$\sum_{t,\ \mathcal{T}_t^{broad} \ni (P_j, P_k)} \alpha_t \times c_{j,k}.$$

To ensure that all the communications can be orchestrated, we have to build a weighted set of matchings and a weighted set of broadcast trees that satisfy the following property: for each edge of the platform graph, the two previous methods for computing its occupation time must give the same result. Therefore, any valid solution of the series of broadcasts problem under the bidirectional one-port model, such that N atomic messages are broadcast in time $Time(N)$, satisfies the following set of equations:

$$\sum_t \alpha_t = N$$

$$\sum_m \beta_m = Time(N)$$

$$\forall (P_j, P_k) \in E,$$

$$\sum_{t,\, \mathcal{T}_t^{broad} \ni (P_j, P_k)} \alpha_t \times c_{j,k} = \sum_{m,\, \mathcal{M}_m^{(B)} \ni (P_j^{out}, P_k^{in})} \beta_m.$$

We normalize all quantities and we rewrite these constraints for $Time(N) = 1$. We build a linear program whose objective function is to maximize the throughput $\rho_{max}^{broad-bi}$, i.e. the (fractional) maximal number of messages that can be broadcast during one time-unit.

Linear program: series of broadcasts

Bidirectional one-port

Maximize $\rho_{max}^{broad-bi} = \sum x_t$

subject to:

$$\sum_m y_m \leq 1$$

$$\forall (P_j, P_k) \in E,$$

$$\sum_{t,\, \mathcal{T}_t^{broad} \ni (P_j, P_k)} x_t \times c_{j,k} = \sum_{m,\, \mathcal{M}_m^{(B)} \ni (P_j^{out}, P_k^{in})} y_m.$$

Consider any valid schedule for the series of broadcasts problem, capable of broadcasting N messages within $Time(N)$ time-units. In this schedule, we denote by α_t the number of messages broadcast using tree $\mathcal{T}_t^{broad}$, and by β_m the overall time when matching $\mathcal{M}_m$ is used to orchestrate communications in this schedule. We set

$$x_t = \frac{\alpha_t}{Time(N)}$$

and

$$y_m = \frac{\beta_m}{Time(N)}.$$

Clearly, x_t and y_m satisfy the constraints of the above linear program. Hence,

$$\sum x_t = \frac{\sum \alpha_t}{Time(N)} = \frac{N}{Time(N)} \leq \rho_{max}^{broad-bi}.$$

Therefore, any valid communication schedule achieving the broadcast of N atomic messages in time $Time(N)$ satisfies

$$\frac{N}{Time(N)} \leq \rho_{max}^{broad-bi}.$$

We conclude that $\rho_{max}^{broad-bi}$ is an upper bound of the achievable throughput.

Conversely, given a solution of the linear program, i.e. a set of weighted matchings and a set of weighted trees such that $\rho_{max}^{broad-bi} = \Sigma x_t$, it is possible to reconstruct a periodic schedule that achieves the optimal throughput $\rho_{max}^{broad-bi}$. This is done using a greedy algorithm which is detailed in Beaumont et al. (2005).

To summarize, if we are able to find the solution of the above linear program, then we can build a schedule with optimal throughput. Section 4 is devoted to the resolution of this linear program in polynomial time. Note that the mere possibility of computing the solution of the linear program (even in rational numbers) in polynomial time is not obvious, since both the number of variables and of constraints may be exponential in the size of the platform graph G: both the number of spanning trees of G and the number of matchings of $G^{(B)}$ may be exponential.

3.1.3 Unidirectional one-port model We adapt the technique developed in the previous section to the unidirectional one-port model. The only change is that simultaneous communications now correspond to a matching in G (denoted by $\mathcal{M}_m^{(G)}$) instead of a matching in the bipartite graph $G^{(B)}$. The linear program obtained is the following:

Linear program: series of broadcasts

Unidirectional one-port

Maximize $\rho_{max}^{broad-uni} = \sum x_t$

subject to:

$$\sum_m y_m \leq 1$$

$$\forall (P_j, P_k) \in E,$$

$$\sum_{t,\, \mathcal{T}_t^{broad} \ni (P_j, P_k)} x_t \times c_{j,k} = \sum_{m,\, \mathcal{M}_m^{(G)} \ni (P_j^{out}, P_k^{in})} y_m.$$

Just as before, we can prove that: (i) the optimal value $\rho_{max}^{broad-uni}$ is an upper bound of the achievable throughput under the unidirectional one-port model; and (ii) a valid schedule achieving the throughput $\rho_{max}^{broad-uni}$ can be reconstructed from the solution of the linear program.

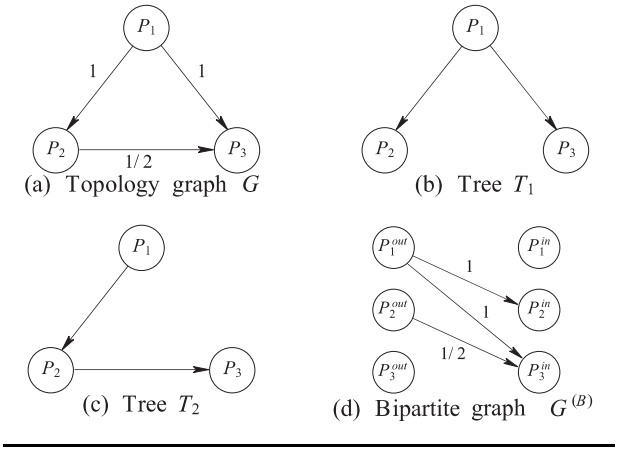

Fig. 3 Example for the broadcast problem: platform topology and two possible broadcast trees.

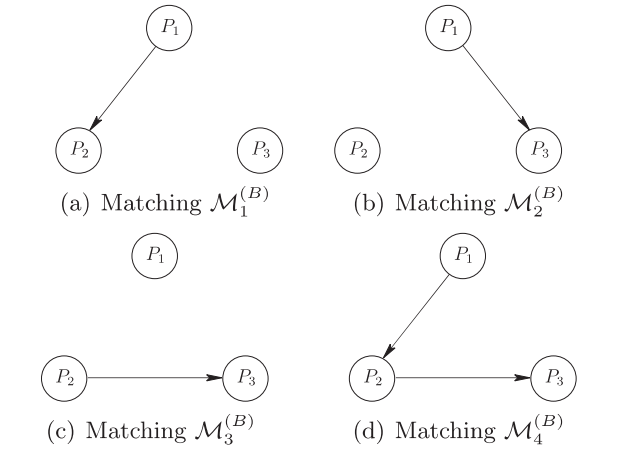

Fig. 4 Possible matchings for the platform depicted in Figure 3(a) under the bidirectional one-port model.

3.1.4 Toy example In order to illustrate the differences between the unidirectional and bidirectional one-port models, as well as the definitions given in the previous sections, we work out a little example on the platform depicted in Figure 3(a), where P_1 broadcasts a message to P_2 and P_3. The set of possible spanning trees is depicted in Figure 3. Consider first that all nodes in the platform operate under the bidirectional one-port model. The bipartite graph $G^{(B)}$ is depicted in Figure 3(d), and the set of matchings in $G^{(B)}$ is depicted in Figure 4.

Therefore, the solution of the following linear system provides both the optimal throughput that can be achieved using the bidirectional one-port model, and the set of trees and matchings used to reach this throughput:

Linear Program–Series of Broadcasts

Bidirectional one-port

Maximize $\rho_{max}^{broad-uni} = \sum x_t$

subject to:

$y_1, y_2, y_3, y_4, x_1, x_2 \geq 0$

$y_1 + y_2 + y_3 + y_4 \leq 1$

$y_1 + y_4 = (x_1 + x_2) \times c_{1,2} = x_1 + x_2$

$y_2 = x_1 \times c_{1,3} = x_1$

$y_3 + y_4 = x_2 \times c_{2,3} = \frac{1}{2} x_2.$

The solution of this linear program is given by $y_4 = x_2 = 1$ and all other variables equal to 0.

Consider now that all the nodes in the platform operate under the unidirectional one-port model. The set of matchings in G is depicted in Figure 5. Therefore, the solution of the following linear system provides both the optimal

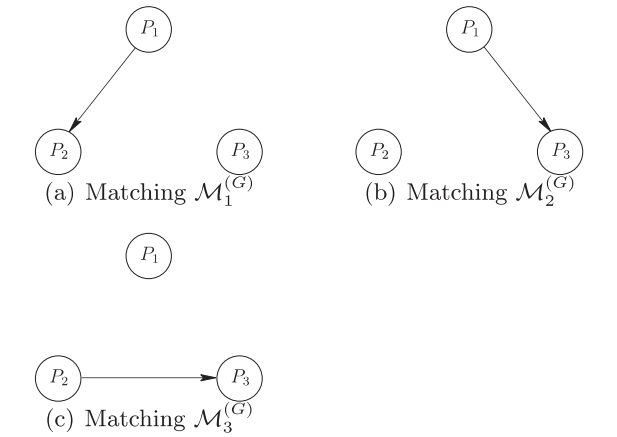

Fig. 5 Possible matchings for the platform depicted in Figure 3(a) under the unidirectional model.

throughput that can be achieved using the unidirectional one-port model, and the set of trees and matchings used to reach this throughput:

Linear program: series of broadcasts

Bidirectional one-port

Maximize $\rho_{max}^{broad-uni} = \sum x_t$

subject to:

$y_1, y_2, y_3, x_1, x_2 \geq 0$

$y_1 + y_2 + y_3 \leq 1$

$y_1 = (x_1 + x_2) \times c_{1,2} = x_1 + x_2$

$y_2 = x_{1,3} \times c_{1,3} = x_1$

$y_3 = x_2 \times c_{2,3} = \frac{1}{2} x_2.$

The solution of this linear program is given by $y_2 = x_1 = (2/3)$, $x_3 = (1/3)$ and all other variables equal to 0 (i.e. T_2 only is used to broadcast the message).

We can check that the best achievable throughput is smaller (2/3) under the unidirectional one-port model than under the bidirectional one-port model (1).

3.2 Series of Multicasts

3.2.1 Formal Definition The formal definition of the series of multicasts problem is very similar to the formal definition of the series of broadcasts problem, even though we will see in Section 3.2.2 that complexity results strongly differ for both problems. As far as the definition is concerned, the main difference is the following. Since the message is sent from P_{source} to the set of destination nodes $\mathcal{D}$, each atomic message must be sent along a tree spanning all the set of nodes in $\mathcal{D}$, but not necessarily all the nodes of the platform.

Let us denote by $\mathcal{T}_1^{mult}$, …, $\mathcal{T}_T^{mult}$ the set of spanning trees of $\mathcal{D}$ (which is finite, even if it may be large) that may be used to broadcast a series of N messages to the set of target processors. We are now looking for a set of weighted spanning trees $(\alpha_1(N), \mathcal{T}_1^{mult})$, …, $(\alpha_T(N), \mathcal{T}_T^{mult})$, where $\alpha_t(N)$ denotes the number of atomic messages (among the N atomic messages) that have been broadcast along the spanning tree $\mathcal{T}_t^{mult}$. The use of the matchings to orchestrate communications is the same as in the case of the series of broadcasts. Consider the following linear programs:

Linear program: series of multicasts

Bidirectional one-port

Maximize $\rho_{max}^{multi-uni} = \sum x_t$

subject to:

$\sum y_m \leq 1$

$\forall (P_j, P_k) \in E,$

$$\sum_{t,\, \mathcal{T}_t^{mult} \ni (P_j, P_k)} x_t \times c_{j,k} = \sum_{m,\, \mathcal{M}_m^{(B)} \ni (P_j, P_k)} y_m.$$

Linear Program–Series of Multicasts

Unidirectional one-port

Maximize $\rho_{max}^{multi-uni} = \sum x_t$

subject to:

$\sum y_m \leq 1$

$\forall (P_j, P_k) \in E,$

$$\sum_{t,\, \mathcal{T}_t^{mult} \ni (P_j, P_k)} x_t \times c_{j,k} = \sum_{m,\, \mathcal{M}_m^{(G)} \ni (P_j^{out}, P_k^{in})} y_m.$$

Then we obtain the same results as in the case of the broadcast operation. Solving these linear programs gives upper bounds on the achievable throughput for the series of multicasts problem, under both the bidirectional or the unidirectional one-port models. Formally, $\rho_{max}^{multi-bi}$ ($\rho_{max}^{multi-uni}$) is an upper bound of the achievable throughput under the bidirectional one-port model (unidirectional one-port model).

Reciprocally, it is possible to build, from the solution of the linear programs, a valid schedule achieving the optimal throughput $\rho_{max}^{multi-bi}$ ($\rho_{max}^{multi-uni}$).

3.2.2 Complexity Results In this section, we derive complexity results for the series of multicasts problem. We first prove that even the simple problem of determining the optimal throughput that can be achieved is NP-hard. Worst, we prove that this optimal throughput cannot be polynomially approximated up to a logarithmic factor (unless P=NP). The interested reader will find all the proofs of these results in the extended version of this paper (Beaumont et al. 2004a).

We formally state the decision problem associated to the determination of the optimal throughput for the series problem. In the following, log denotes the logarithm in base 2.

Definition 1. (COMPACT-MULTICAST). Given a weighted platform graph $G = (P, E, c)$, a source processor P_{source}, a set of destination processors $\mathcal{D}$, a rational bound for the throughput ρ, and a rational bound for the size S, is there a K-periodic schedule of period T, i.e. a schedule which performs K multicast operations every T units of time in steady-state, such that $K \leq \log S$ and $\frac{K}{T} \geq \rho$?

Theorem 1. *COMPACT-MULTICAST(G, P_{source}, $\mathcal{D}$, ρ, S) is NP-complete.*

We point out that the bound S is introduced so that the description of a periodic schedule can be polynomial in the problem size. Informally, a K-periodic schedule is the superposition of K multicast trees, and the condition $K \leq \log S$ ensures that all these trees can be encoded with a size polynomial in the input. Each tree is at most the size of the platform graph, and there are no more than $\log S$ of them. We point out that similar difficulties hold for specifying cyclic schedules in general: see the survey paper of Hanen and Munier (1995).

The proof of this result (available in Beaumont et al. 2004a) can be used to derive an inapproximability result. The class APX is defined as the problems in NP which admit a polynomial-time λ-approximation algorithm, for some constant λ. Therefore, if we show that COMPACT-MULTICAST does not belong to this class, this will prove that, unless P=NP, no polynomial-time heuristic can approximate the best throughput, up to an arbitrary constant factor.

Theorem 2. *COMPACT-MULTICAST does not belong to the class APX.*

We can refine Theorem 1 by suppressing the restriction on the compactness of the solution. We first come to a formulation of the problem using weighted multicast trees.

Definition 2. (COMPACT-WEIGHTED-MULTICAST). Given a weighted platform graph $G = (P, E, c)$, a source processor P_{source}, a set of destination processors $\mathcal{D}$, a rational bound for the throughput ρ, is there a periodic schedule consisting of $k \leq 2|E|$ multicast trees $\{T_1, \ldots, T_k\}$, where α_i is the average number of messages sent through tree T_i within one time-unit, $\alpha_i = a_i/b_i$, where a_i and b_i are integers such that $\forall i = 1, \ldots, k$, $\log a_i + \log b_i \leq 4|E|(\log|E| + A)$, where $A = \log \max c_{i,j}$), and $\Sigma\alpha_i \geq \rho$?

Theorem 3. *COMPACT-WEIGHTED-MULTICAST(G, P_{source}, $\mathcal{D}$, ρ, S) is NP-complete.*

The following result states that restricting to compact weighted trees does not affect the optimality of the solution.

Theorem 4. *Given a weighted platform graph $G = (P, E, c)$, a source processor P_{source}, a set of destination processors $\mathcal{D}$, if there exists a periodic schedule that achieves a throughput ρ, then there also exists a solution of COMPACT-WEIGHTED-MULTICAST(G, P_{source}, $\mathcal{D}$, ρ).*

Although the series of multicasts leads to NP-complete problems, the series of broadcasts is more tractable. To solve it, we need to introduce powerful mathematical tools that will enable us to solve the corresponding linear programs.

4 Ellipsoid Method and the Series of Broadcasts Problem

4.1 Basics on the Ellipsoid Method

In order to prove that series of broadcasts problems can be solved in polynomial time under both the unidirectional and bidirectional models, we first need to introduce a few concepts related to the ellipsoid method for solving linear programs (Khachiyan 1979). We rely on sophisticated techniques to solve the linear programs introduced in Section 3 because, as already pointed out, both the number of constraints and the number of variables may be exponential in the size of the problem (i.e. the size of the graph G). The interested reader may refer to Grötschel, Lovász, and Schrijver (1994) to find more details on the ellipsoid method and the proof of the results used in this section. See also Beaumont and Marchal (2004) to find a more detailed presentation of the resolution of the series of broadcasts problem under the unidirectional one-port model.

Consider a convex polyhedron $\mathcal{P}$ defined by a finite set of inequalities. We can define the following two optimization problems on this convex.

Definition 3. (The strong optimization problem (SOPT($\mathcal{P}$, C))). Given a convex $\mathcal{P}$ and a vector $C \in \mathbb{Q}^n$, find a vector $x \in \mathcal{P}$ that maximizes $C^T \cdot x$ or assert that $\mathcal{P}$ is empty.

Definition 4. (The strong separation problem (SSEP ($\mathcal{P}$, x))). Given a vector $x \in \mathbb{Q}^n$, decide whether $x \in \mathcal{P}$, and if not, find a vector hyperplane that separates x from $\mathcal{P}$; more exactly, find a vector $C \in \mathbb{Q}^n$ such that $C^T \cdot x > \max\{C^T \cdot y \mid y \in \mathcal{P}\}$.

We proved in Section 3 that the series of broadcasts problem (under both communication models) can be expressed as a strong optimization problem. The ellipsoid theory basically says that if we are able to solve the strong sepa-

ration problem (which is usually much easier) in polynomial time, then we are also able to solve the strong optimization problem in polynomial time. Formally, this property is proved by the following theorem (Grötschel, Lovász, and Schrijver 1994, chapter 6).

Theorem 5. (Theorem 6.4.9 in Grötschel, Lovász, and Schrijver (1994)). *Any one of the following two problems*

- *strong separation*
- *strong optimization*

can be solved in oracle-polynomial time for any well-described polyhedron given by an oracle for the other problem.

The proof that polyhedrons presented in Section 3 are well described can be found in Beaumont and Marchal (2004). Then, the following theorem proves that it is possible to build a valid solution of the initial optimization problem from the execution of the ellipsoid method on the dual optimization problem.

Theorem 6. (Theorem 6.5.15 in Grötschel, Lovász, and Schrijver (1994)). *There exists an oracle-polynomial time algorithm that, for any $c \in \mathbb{Q}^n$ and for any well-described polyhedron $(\mathcal{P}; n, \phi)$ given by a strong separation oracle where every output has an encoding length at most ϕ, either*

(i) finds a basic optimum dual solution with oracle inequalities, or
(ii) asserts that the dual problem is unbounded or has no solution.

4.2 Applications to the Series of Broadcast Problems

In order to apply the ellipsoid method, we need to consider the dual problems of the optimization problems that have been introduced in Section 3. Let us denote by $e_1, \ldots, e_{|E|}$ the set of edges in $G = (P, E)$.

The dual formulation of the series of broadcasts problems under the unidirectional and bidirectional models are, respectively:

Dual Formulation

Unidirectional one-port model

MINIMIZE z_1,

SUBJECT TO

$$(\mathcal{D}_{uni}) \begin{cases} z_1 \geq 0 \\ \forall \mathcal{M}_m^{(G)} \quad \sum_{e_k \in \mathcal{M}_m^{(G)}} z_{k+1} \leq z_1 \\ \forall \mathcal{T}_t^{broad} \quad \sum_{e_k = (P_i, P_j) \in \mathcal{T}_t^{broad}} c_{i,j} \cdot z_{k+1} \geq 1 \end{cases}$$

and

Dual Formulation

Bidirectional one-port model

MINIMIZE z_1,

SUBJECT TO

$$(\mathcal{D}_{bi}) \begin{cases} z_1 \geq 0 \\ \forall \mathcal{M}_m^{(B)} \quad \sum_{e_k \in \mathcal{M}_m^{(B)}} z_{k+1} \leq z_1 \\ \forall \mathcal{T}_t^{broad} \quad \sum_{e_k = (P_i, P_j) \in \mathcal{T}_t^{broad}} c_{i,j} \cdot z_{k+1} \geq 1 \end{cases}$$

The strong separation problems associated with the dual formulation under the bidirectional and unidirectional one-port models are the following.

Definition 5. (Broadcast-SSEP-Bidirectional (Unidirectional)(G, P_{source}, x)). Let G denote the platform graph, let P_{source} be the source node and $z \in \mathbb{Q}^{|E|+1}$ is a vector $C \in \mathbb{Q}^{|E|+1}$ such that $C^T \cdot z > \max\{C^T \cdot y, y \in \mathcal{D}_{bi}(\mathcal{D}_{uni})\}$.

Lemma 1. *Broadcast-SSEP-Bidirectional (G, P_{source}, z) and Broadcast-SSEP-Unidirectional (G, P_{source}, z) can be solved in polynomial time.*

Proof. Consider an instance z of the strong separation problem $\mathcal{D}_{bi}$ (or $\mathcal{D}_{uni}$). In this case, solving the strong separation problem consists in checking (in polynomial time) if all the constraints are satisfied, and, if this is not the case, to exhibit a violated constraint. In both models, we can classify the constraints into three classes, and exhibit a polynomial method to check if all constraints of a given class are satisfied:

$$\begin{cases} \text{Constraint } I: \\ \qquad z_1 \geq 0 \\ \text{Constraints } IIa \text{ (unidirectional model):} \\ \qquad \forall \mathcal{M}_m^{(G)} \sum_{e_k \in \mathcal{M}_m^{(G)}} z_{k+1} \leq z_1 \\ \text{Constraints } IIb \text{ (bidirectional model):} \\ \qquad \forall \mathcal{M}_m^{(B)} \sum_{e_k \in \mathcal{M}_m^{(B)}} z_{k+1} \leq z_1 \\ \text{Constraints } III: \\ \qquad \forall \mathcal{T}_t^{broad} \sum_{e_k = (P_i, P_j) \in \mathcal{T}_t^{broad}} c_{i,j} \cdot z_{k+1} \geq 1 \end{cases}$$

- Constraint *I*: z being given, constraint *I* can clearly be checked in polynomial time.
- Constraints *IIa*:

$$\forall \mathcal{M}_m^{(G)} \sum_{e_k \in \mathcal{M}_m^{(G)}} z_{k+1} \leq z_1 \; (bidirectional\ model)$$

  Consider the weighted graph $G_{\mathcal{M}} = (P, E, c_{\mathcal{M}})$, where $c_{\mathcal{M}}(e_k) = z_{k+1}$. A matching $M_{max}^{\mathcal{M}}$ of maximal weight $w_{max}^{\mathcal{M}}$ can be computed in $G_{\mathcal{M}}$ in polynomial time (Cormen, Leiserson, and Rivest 1990). If $w_{max}^{\mathcal{M}} \leq z_1$, then all constraints *IIa* are satisfied, and if $w_{max}^{\mathcal{M}} > z_1$ then the constraint *IIa* corresponding to matching $M_{max}^{\mathcal{M}}$ is violated. We are therefore able to solve the strong separation problem for constraints *IIa* in polynomial time.
- Constraints *IIb*

$$\forall \mathcal{M}_m^{(B)} \sum_{e_k \in \mathcal{M}_m^{(B)}} z_{k+1} \leq z_1 (unidirectional\ model)$$

  Consider the weighted bipartite graph $G_{\mathcal{M}}^{(B)} = (G^{(B)}, c_{\mathcal{M}})$, where $c_{\mathcal{M}}(e_k) = x_{k+1}$. A matching $M_{max}^{\mathcal{M}}$ of maximal weight $w_{max}^{\mathcal{M}}$ can be computed in $G_{\mathcal{M}}$ in polynomial time (Cormen, Leiserson, and Rivest 1990). If $w_{max}^{\mathcal{M}} \leq z_1$, then all constraints *IIb* are satisfied; if $w_{max}^{\mathcal{M}} > z_1$, then the constraint *IIb* corresponding to matching $M_{max}^{\mathcal{M}}$ is violated. We are therefore able to solve the strong separation problem for constraints *IIb* in polynomial time.
- Constraints *III*:

$$\forall \mathcal{T}_t^{broad} \sum_{e_k = (P_i, P_j) \in \mathcal{T}_t^{broad}} c_{i,j} \cdot z_{k+1} \geq 1$$

  In this equation $\mathcal{T}_t^{broad}$ denotes a spanning tree of G. Consider the following graph $G_T = (P, E, c_T)$, where the weight of an edge $e_k = (P_i, P_j)$ is $c_T(e_k) = c_{i,j} \cdot z_{k+1}$. A tree $T_{min}^{\mathcal{T}}$ of minimal weight $w_{min}^{\mathcal{T}}$ can be computed in G_T in polynomial time (Cormen, Leiserson, and Rivest 1990). If $w_{min}^{\mathcal{T}} \geq 1$, then all constraints *III* are satisfied, and if $w_{min}^{\mathcal{T}} < 1$, then the constraint *III* corresponding to tree $T_{min}^{\mathcal{T}}$ is violated. We are therefore able to solve the strong separation problem for constraints *III* in polynomial time.

Therefore, we are able, for any z, to assert in polynomial time (for both the bidirectional and unidirectional one-port models) if z satisfies the constraints and if it does not, to give at least a violated constraint. Thus, we are able to solve the strong separation problem either in $\mathcal{D}_{bi}$ (or $\mathcal{D}_{uni}$).

Then, using Lemma 1 and Theorem 6, we can prove the following theorem.

Theorem 7. *Let G represent a platform graph and a given source node $P_{source} \in G$. The series of broadcasts problem can be solved in polynomial time either under the bidirectional or the unidirectional one-port models.*

5 Extensions

5.1 Using the Ellipsoid Method

The technique presented in Section 4 is very efficient in order to prove that several series versions of well-known collective communication problems can be solved in polynomial time, either under the unidirectional or bidirectional one-port models.

We do not detail any proof in this section (instead we point to bibliographical references), but rather indicate how the general framework of Section 4 applies to several other collective communication schemes.

- *Series of Scatters*. In the series of scatters problem, a source node P_{source} initially holds a large set of atomic messages to send to some target (destination) nodes. The main difference between scatter and multicast is that the messages to be sent are different for each destination node. If we apply the framework described in Section 4, the strong separation problem consists of:
  - finding a matching of maximal weight in G (unidirectional one-port model) or $G^{(B)}$ (bidirectional one-port model) (problems already solved for the broadcast problem);
  - finding a path of minimal weight in G.

The determination of a path of minimal weight can be solved in polynomial time (Cormen, Leiserson, and Rivest 1990), and therefore the series of scatters problem can be solved in polynomial time.

- *Series of Gathers*. In the series of gathers problem, each node successively sends (different) messages to a common destination node P_{dest}. In the framework described in Section 4, the series of gathers and series of scatters problems are equivalent, and both require the determination of a matching of maximal weight and a path of minimal weight. The series of gathers problem can therefore be solved in polynomial time.
- *Series of Total Exchanges*. Each processor initiates a series of broadcasts, and the series of all-to-alls, where each processor initiates a series of scatters, can also be solved in polynomial time using the same framework (see Beaumont and Marchal 2004).

- *Series of Multicasts*. This problem can also be tackled using the framework described in Section 4, but this approach does not lead to a polynomial algorithm (which is not surprising since we have proved in Section 3.2.2 that the series of multicasts problem is NP-complete). In fact, the strong separation problem for the series of multicasts problem consists of:

  – finding a matching of maximal weight in G (unidirectional one-port model) or $G^{(B)}$ (bidirectional one-port model), which can be solved in polynomial time;
  – finding a Steiner tree (a tree that spans the node in $\mathcal{D}$ only) of minimal weight in G. This problem is known to be NP-complete (Garey and Johnson 1979).

  The property that the strong separation problem is NP-complete is not enough to prove the NP-completeness of the series of multicasts problem, but we have provided a proof of this result in Section 3.2.2.

5.2 Extending NP-Completeness Results

In Beaumont et al. (2004a), it is proved that the series of parallel prefix computations problem is NP-complete. The proof uses the framework described in Section 3.2.2. In the series of parallel prefix computations problem, each node P_i holds a series of contributions. At the end, each node P_i must hold the sum of all the contributions of processors whose index is smaller than or equal to i. It is worth pointing out that the series of reduce operations problem, where only node P_0 must hold the sum of the contributions of other nodes, can be solved in polynomial time (Legrand, Marchal, and Robert 2003).

5.3 Efficient Polynomial Algorithms

The framework described in Section 4 is very convenient in order to determine whether a series of collective communications problem can be solved in polynomial time. Nevertheless, the algorithm that we propose relies on the ellipsoid method (Khachiyan 1979) to solve linear programs. Therefore, it does not lead to efficient polynomial algorithms, since in practice the execution time of the ellipsoid method is huge (though polynomial) (Grötschel, Lovász, and Schrijver 1994).

Under the bidirectional one-port model, efficient polynomial algorithms are known for the following problems: series of broadcasts (Beaumont et al. 2005), series of reduce operations and series of scatters (Legrand, Marchal, and Robert 2003). The latter algorithm can be easily adapted for the series of gathers, series of total exchanges and series of all-to-alls problems. However, to the best of our knowledge, no efficient polynomial algorithm is known for any of these problems under the unidirectional one-port model.

6 Conclusion

In this paper, we have provided a unified framework for assessing the complexity of determining the optimal throughput of series of collective communications problems: either we provided a solution in polynomial time, or we provided NP-completeness results (e.g. for the multicast scheme, as in Section 3.2.2). We have used the realistic communication models presented in Section 2.

However, the general method presented in this paper relies on the ellipsoid method for solving linear programs (explained in Section 4) and therefore does not lead to efficient polynomial algorithms. Efficient algorithms have been developed for most collective communication schemes under the bidirectional one-port model. The design of efficient algorithms under the unidirectional one-port model is still an open problem.

When designing and implementing realistic collective communications on heterogeneous distributed platforms, another key issue is the robustness, i.e. the ability to handle small variations in link performances. This issue has not been dealt with yet, but further work should try to consider it, and to extend current approaches so as to derive robust communication primitives.

Author Biographies

Olivier Beaumont obtained his Ph.D. from the University of Rennes 1 in 1999. Since then, he has been appointed as Assistant Professor at ENS Lyon (2000–2002) and at the University of Bordeaux (2002–2005). He obtained his "Habilitation a Diriger les Recherches" from the Uni-

versity of Bordeaux 1 in 2004. He is the author of 15 papers published in international journals and more than 40 conference papers. His main interests are distributed computing, parallel algorithms, grid and peer-to-peer systems.

Loris Marchal received his Master's degree from Ecole Normale Superieure de Lyon in 2003. He is currently a Ph.D. student in the LIP laboratory at ENS Lyon. He is mainly interested in parallel algorithm design for heterogeneous platforms and in scheduling techniques.

Yves Robert received his Ph.D. degree from Institut National Polytechnique de Grenoble in 1986. He is currently a full professor in the Computer Science Laboratory LIP at ENS Lyon. He is the author of four books, 95 papers published in international journals, and 115 papers published in international conferences. His main research interests are scheduling techniques and parallel algorithms for clusters and grids. He is a senior member of IEEE and the IEEE Computer Society, and serves as an associate editor of *IEEE Transactions on Parallel and Distributed Systems*.

References

Anastasi, G., Bartoli, A., and Spadoni, F. 2001. A reliable multicast protocol for distributed mobile systems: design and evaluation. *IEEE Transactions on Parallel Distributed Systems* 12(10):1009–1022.

Beaumont, O. and Marchal, L. 2004. Pipelining broadcasts on heterogeneous platforms under the one-port model. Research Report 2004-32, LIP, ENS Lyon, France, July.

Beaumont, O., Legrand, A., Marchal, L., and Robert, Y. 2004a. Complexity results and heuristics for pipelined multicast operations on heterogeneous platforms. Research Report RR-2004-07, LIP, ENS Lyon, France, January.

Beaumont, O., Legrand, A., Marchal, L., and Robert, Y. 2004b. Assessing the impact and limits of steady-state scheduling for mixed task and data parallelism on heterogeneous platforms. Research Report RR-2004-20, LIP, ENS Lyon, France, April.

Beaumont, O., Legrand, A., Marchal, L., and Robert, Y. 2005. Pipelining broadcasts on heterogeneous platforms. *IEEE Transactions on Parallel Distributed Systems* 16(4):300–313. Also available as LIP Research Report 2003-34.

Cohen, J., Fraigniaud, P., König, J., and Raspaud, A. 1998. Optimized broadcasting and multicasting protocols in cut-through routed networks. *IEEE Transactions on Parallel Distributed Systems* 9(8):788–802.

Cormen, T. H., Leiserson, C. E., and Rivest, R. L. 1990. *Introduction to Algorithms*, MIT Press, Cambridge, MA.

Desprez, F., Fraigniaud, P., and Tourancheau, B. 1993. Successive broadcasts on Hypercube. Technical Report CS-93-210, University of Tennessee, Knoxville.

Garey, M. R. and Johnson, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco.

Gopalsamy, T., Singhal, M., Panda, D., and Sadayappan, P. 2002. A reliable multicast algorithm for mobile ad hoc networks. *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, Vienna, Austria, July 2–5, pp. 563–570.

Grötschel, M., Lovász, L., and Schrijver, A. 1994. *Geometric Algorithm and Combinatorial Optimization. Algorithms and Combinatorics 2*, 2nd corrected edition, Springer-Verlag, Berlin.

Hanen, C. and Munier, A. 1995. Cyclic scheduling on parallel processors: an overview. *Scheduling Theory and its Applications*, P. Chrétienne, E. G. Coffman, J. K. Lenstra, and Z. Liu, editors, Wiley, New York, pp. 193–226.

Hwang, K. and Xu, Z. 1998. *Scalable Parallel Computing*, McGraw-Hill, New York.

Johnsson, S. L. and Ho, C.-T. 1989. Optimum broadcasting and personalized communication in hypercubes. *IEEE Transactions on Computers* 38(9):1249–1268.

Khachiyan, L. G. 1979. A polynomial algorithm in linear programming (in Russian). *Soviet Mathematikcs Doklady* 20:191–194.

Ko, H., Latifi, S., and Srimani, P. 2000. Near-optimal broadcast in all-port wormhole-routed hypercubes using error-correcting codes. *IEEE Transactions on Parallel and Distributed Systems* 11(3):247–260.

Kumar, K. and Jaffe, J. 1983. Routing to multiple destinations in computer networks. *IEEE Transactions on Communications* 31(3):343–351.

Kumar, V., Grama, A., Gupta, A., and Karypis, G. 1994. *Introduction to Parallel Computing*, Benjamin-Cummings, New York.

Legrand, A., Marchal, L., and Robert, Y. 2003. Optimizing the steady-state throughput of scatter and reduce operations on heterogeneous platforms. Technical Report RR-2003-33, LIP, ENS Lyon, France, June.

Lin, X. and Ni, L. 1993. Multicast communication in multicomputer networks. *IEEE Transactions on Parallel Distributed Systems* 4(10):1105–1117.

Moore, J. and Quinn, M. 1997. Generating an efficient broadcast sequence using reflected gray codes. *IEEE Transactions on Parallel and Distributed Systems* 8(11):1117–1122.

Robinson, D., McKinley, P., and Cheng, B. 1995. Optimal multicast communication in wormhole-routed torus networks. *IEEE Transactions on Parallel Distributed Systems* 6(10):1029–1042.

Sivaram, R., Kesavan, R., Panda, D., and Stunkel, C. 2001. Architectural support for efficient multicasting in irregular networks. *IEEE Transactions on Parallel Distributed Systems* 12(5):489–513.

Snir, M., Otto, S. W., Huss-Lederman, S., Walker, D. W., and Dongarra, J. 1996. *MPI: The Complete Reference*, MIT Press, Cambridge, MA.

Tseng, Y-C., Wang, S-Y., and Ho, C-W. 1999. Efficient broadcasting in wormhole-routed multicomputers: a network-partitioning approach. *IEEE Transactions on Parallel and Distributed Systems* 10(1):44–61.

Wang, S-Y. and Tseng, Y-C. 2000. Algebraic foundations and broadcasting algorithms for wormhole-routed all-port tori. *IEEE Transactions on Computers* 49(3):246–258.

Watts, J. and Van De Geijn, R. 1995. A pipelined broadcast for multidimensional meshes. *Parallel Processing Letters* 5(2):281–292.

Wieselthier, J., Nguyen, G., and Ephremides, A. 2002. Energy-aware wireless networking with directional antennas: the case of session-based broadcasting and multicasting. *IEEE Transactions on Mobile Computing* 1(3):176–191.

Winter, P. 1987. Steiner problem in networks: a survey. *Networks* 17(2):129–167.

Yang, Y., Wang, J., and Qiao, C. 2000. Non-blocking WDM multicast switching networks. *IEEE Transactions on Parallel Distributed Systems* 11(12):1274–1287.