Laboratoire de l'Informatique du Parallélisme

École Normale Supérieure de Lyon
Unité Mixte de Recherche CNRS-INRIA-ENS LYON-UCBL n° 5668

Pipelining Broadcasts on Heterogeneous Platforms under the One-Port Model

Olivier Beaumont,
Loris Marchal

July 2004

Research Report N° 2004-32

École Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : +33(0)4.72.72.80.37
Télécopieur : +33(0)4.72.72.80.80
Adresse électronique : `lip@ens-lyon.fr`

CENTRE NATIONAL
DE LA RECHERCHE
SCIENTIFIQUE

INRIA

Pipelining Broadcasts on Heterogeneous Platforms under the One-Port Model

Olivier Beaumont, Loris Marchal

July 2004

Abstract

In this paper, we consider the communications involved by the execution of a complex application, deployed on a heterogeneous platform. Such applications extensively use macro-communication schemes, for example to broadcast data items. Rather than aiming at minimizing the execution time of a single broadcast, we focus on the steady-state operation. We assume that there is a large number of messages to be broadcast in pipeline fashion, and we aim at maximizing the throughput, i.e. the (rational) number of messages which can be broadcast every time-step. We target heterogeneous platforms, modeled by a graph where resources have different communication speeds under the unidirectional one-port model (i.e. at a given time step, a processor can be involved in at most one (incoming or outgoing) communication with one of its neighbors). Achieving the best throughput may well require that the target platform is used in totality: we show that neither spanning trees nor DAGs are as powerful as general graphs.

We propose a rather sophisticated polynomial algorithm for determining the optimal throughput that can be achieved using a platform, together with a (periodic) schedule achieving this throughput. The algorithm is based on the use of polynomial oracles and of the ellipsoid method [9, 13] for solving in linear programs in rational numbers. The polynomial compactness of the description comes from the decomposition of the schedule into several broadcast trees that are used concurrently to reach the best throughput. It is important to point out that a concrete scheduling algorithm based upon the steady-state operation is asymptotically optimal, in the class of all possible schedules (not only periodic solutions).

Keywords: Scheduling, steady-state, heterogeneous platforms, ellipsoid method

Résumé

Nous nous intéressons ici aux communications qui ont lieu lors de l'exécution d'une application complexe distribuée sur un environnement hétérogène de type "*grille de calcul*". De telles applications font un usage intensif de la diffusion de données à travers le réseau d'interconnexion. Nous cherchons à optimiser le débit d'une telle diffusion en régime permanent, en supposant qu'un grand nombre de messages doivent être diffusés successivement, comme c'est le cas pour le parallélisme de données. Nous visons des plates-formes hétérogènes, modélisées par un graphe où les ressources de communications ont des vitesses différentes et utilisent le modèle un-port ; i.e. à un instant donné, un nœud est impliqué dans au plus un transfert de données (émission ou réception). Nous étudions la diffusion utilisant des sous-réseaux avec des topologies restreintes (arbres ou graphes acycliques dirigés) et montrons que celles-ci sont moins puissantes que des graphes généraux. Nous proposons un algorithme sophistiqué pour calculer le débit optimal dans le cas général et construire un ordonnancement périodique qui le réalise. Notre algorithme est basé sur l'utilisation d'oracles polynomiaux et de la méthode de l'ellipsoïde [9, 13] afin de résoudre des systèmes linéaires en nombres rationnels. Nous montrons que l'ordonnancement réalisé est asymptotiquement optimal, parmi tous les ordonnancements possibles (pas seulement les solutions périodiques).

Mots-clés: Ordonnancement, régime permanent, plates-formes hétérogènes, méthode de l'ellipsoïde

1 Introduction

Broadcasting in computer networks is the focus of a vast literature. The one-to-all broadcast, or single-node broadcast [16], is the most primary collective communication pattern: initially, only the source processor has the data that needs to be broadcast; at the end, there is a copy of the original data residing at each processor.

Parallel algorithms often require to send 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 [12, 29, 27, 15, 28]). The *one-to-all* MPI routine [26] is widely used, and particular case has been given to its efficient implementation on a large variety of platforms [11]. There are three main variants considered in the literature:

Atomic broadcast: 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 network.

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

Series of broadcasts: the same source processor sends a series of atomic one-to-all broadcasts, involving messages of the same size. The processing of these broadcasts can be pipelined.

For the first two problems, the goal is to minimize the total execution time (or makespan). 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 per time-unit.

In the case of the *atomic broadcast*, there is no reason why a processor (distinct from the source) would receive the message twice. Therefore, the atomic broadcast is frequently implemented using a spanning tree. In the case of the *pipelined broadcast*, things get more complex: the idea is to use several edge-disjoint spanning trees to route simultaneously several fractions of the total message. Along each spanning tree, the message fraction is divided into packets, which are sent in a pipeline fashion, so as to minimize start-up idle times. See [29] for an illustration with two-dimensional meshes.

The *series of broadcasts* problems has been considered by Moore and Quinn [22], and by Desprez et al. [8], 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 *series of broadcasts* resembles the *pipelined broadcast* problem in that we can solve the latter using an algorithm for the former: this amounts to fix the granularity, i.e. the size of the atomic messages (packets) that will be sent in pipeline. However, an efficient solution to the *pipelined broadcast* problem would require to determine the size of the packets as a function of the total message length.

The *series of broadcasts* problem has already been addressed in the case of heterogeneous computing platforms in [4], under the bidirectional one-port model, where a node can simultaneously send a message (and only one) to one of its neighbors and receive a message (and only one) from one of its neighbors. In [4], a polynomial algorithm is proposed to determine the optimal throughput and a schedule achieving this throughput, using network flows and graph theory (Edmond's branching theorem). We will prove in Section 2.2.2 that under the unidirectional 1-port model, we need to rely on more sophisticated algorithmic techniques, since we cannot consider separately the problem of finding the set of trees used to broadcast the message and the problem of building a schedule based on this set of trees. In this paper, we propose an algorithm for computing simultaneously the set of trees and the schedule. This algorithm relies on tools such as rational linear programming (ellipsoid method with polynomial oracles) and fractional edge coloring for general graphs.

The rest of the paper is organized as follows. The next section (Section 2) is devoted to the formal specification of our broadcast problems and of the target heterogeneous network (2.1). In particular, we underline the differences between the unidirectional and the bidirectional one-port model and how this affects the design of an algorithm for the *series of broadcasts* problem (2.2). In 2.3, we compare topologies for the *series of broadcasts* problem and we prove that general platforms are strictly more powerful than DAGs, and that DAGs are strictly more powerful than trees. In Section 3, we move to the design of the optimal steady-state algorithm, when the target network is a general graph. We extend this study to other collective communication scheme in Section 4, and to the case of a complex platform with mixed one-port and multi-port nodes in Section 5. In 6, we prove that our algorithm (which optimize the throughput) is asymptotically optimal if we try to optimize the makespan. We briefly survey related work in Section 7, and we state some concluding remarks in Section 8.

2 Framework

2.1 Platform model

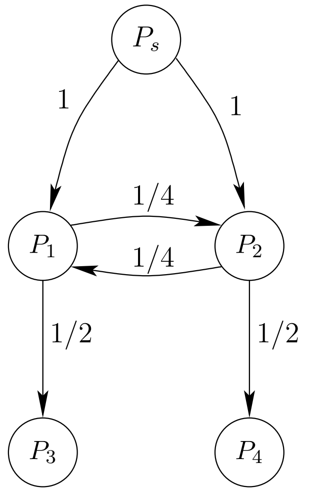


Figure 1: Simple network topology. The value of $c_{j,k}$ is indicated along each edge. The node P_s is the source of the broadcasts.

The target architectural platform is represented by an edge-weighted directed graph $G = (V, E, c)$, as illustrated in Figure 1. Note that this graph may well include cycles and multiple paths. Let $p = |V|$ be the number of nodes. There is a *source* node P_s, which plays a particular role: it initially holds all the data to be broadcast. All the other nodes P_i, $1 \leqslant i \leqslant p, i \neq s$, are destination nodes which must receive all the data sent by P_s.

There are several scenarios for the operation of the processors, which will be discussed in Section 7. In this paper, we concentrate on the *unidirectional one-port model*, where a processor node can receive data from one of its neighbor or send data to one of its neighbor. At any given time-step, there is at most one communication involving a given processor, one in emission or one in reception. We will discuss in next section the differences induced by the *unidirectional one-port model* with respect to the *bidirectional one-port model*, where at a given time-step, there are at most two communications involving a given processor, one in emission and one in reception.

Each edge $e_{j,k} : P_j \to P_k$ is labelled by a value $c_{j,k}$ which represents the time needed to communicate one unit-size message from P_j to P_k (start-up costs are dealt with below, for the *pipelined broadcast* problem). 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. We state the communication model more precisely: if P_j sends a unit-size message to P_k at time-step t, then (i) P_k cannot initiate another receive operation or another send operation before time-step $t + c_{j,k}$, and (ii) P_j cannot initiate another send operation or another receive operation before time-step $t + c_{j,k}$.

Series of broadcasts In the *series of broadcasts* problem, the source processor broadcasts a (potentially infinite) sequence of unit-size messages. Start-up costs are included in the values of the link capacities $c_{j,k}$. The optimization problem $\textsc{Series}(V, E, c)$ is to maximize the throughput.

2.2 Fundamental differences between unidirectional and bidirectional one-port models

2.2.1 Bidirectional one-port model

In order to describe the algorithm in the case of unidirectional one port nodes and to illustrate the differences between unidirectional and bidirectional one-port models, we work out the little example of the platform depicted in Figure 2, where P_1 broadcasts a message to P_2 and P_3. We first briefly summarize the method given in [4] in the case of the bidirectional one-port model.

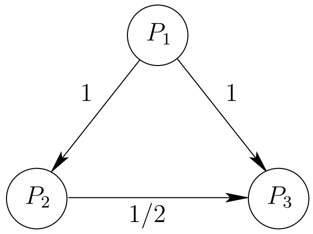


Figure 2: Example of topology for the broadcast problem

In what follows, we try to determine the maximal fractional number of messages a that can be broadcast during one time unit, at steady state. Let us denote by $x_i^{j,k}$ the (fractional) number of messages sent to P_i which transit on the edge between P_j and P_k (in our example, we only need to consider $x_2^{1,2}, x_3^{1,2}, x_3^{1,3}$ and $x_3^{2,3}$. We express some constraints one these quantities with linear inequalities or equalities. The first set of equalities says that the whole message must leave P_1 and must reach P_2 and P_3:

$$\begin{cases} x_2^{1,2} = a & \text{(all messages targeting } P_2 \text{ leave } P_1) \\ x_3^{1,2} + x_3^{1,3} = a & \text{(all messages targeting } P_3 \text{ leave } P_1) \\ x_2^{1,2} = a & \text{(all messages targeting } P_2 \text{ reach } P_2) \\ x_3^{1,3} + x_3^{2,3} = a & \text{(all messages targeting } P_3 \text{ reach } P_3) \end{cases}$$

We also need to add a conservation law, which states that all the messages whose final destination is P_i and that arrive in $P_j \neq P_i$ must leave P_j. In our example, the only interesting case is the set of messages whose final destination is P_3 and that transit by P_2:

$$x_3^{1,2} = x_3^{2,3}.$$

The following set of constraints is related to link occupation. The fraction of messages that transit on (P_j, P_k) are $x_1^{j,k}, x_2^{j,k}, \dots, x_n^{j,k}$. Since we consider a broadcast (and not a scatter), some of the messages with different final destination may well be identical. In this case, $x_i^{j,k}$ and $x_{i'}^{j,k}$ may count for the same message. Thus the link between P_j and P_k may well be busy during less than $c_{j,k} \times \sum_i x_i^{j,k}$ time units (remind that $c_{j,k}$ is the time necessary to transfer one single message). In [4], it is proved that on the contrary, it is always possible to organize the communications so that all the messages transiting between P_j and P_k are sub-messages of the largest one, so that if $t_{j,k}$ denotes the fraction of time when the link between P_j and P_k is busy, then

$$t_{j,k} = c_{j,k} \times \max_i x_i^{j,k} \text{ holds true.}$$

For our example, it leads to the following set of inequalities

$$t_{1,2} \leqslant 1 \times x_2^{1,2}, \quad t_{1,2} \leqslant 1 \times x_3^{1,2}, \quad t_{1,3} \leqslant 1 \times x_3^{1,3}, \quad t_{2,3} \leqslant \frac{1}{2} \times x_3^{2,3}.$$

The last set of inequalities is related to port occupation. In the bidirectional one-port model, we assume one-port for incoming communications and one-port for outgoing communications so that, if we denote by t_j^{in} and t_j^{out} the occupation time of the communication ports of P_j, we obtain, for our example

$$\begin{cases} t_1^{out} = t_{1,2} + t_{1,3} \\ t_2^{in} = t_{1,2} \\ t_2^{out} = t_{2,3} \\ t_3^{in} = t_{2,3} + t_{1,3} \end{cases}$$

At last, since our aim is to maximize the number of messages broadcast during one time unit, none of incoming or outgoing ports can be used more than one unit of time and we thus obtain the following linear program

Bidirectional Steady-State Broadcast Problem on G
Maximize a,
subject to

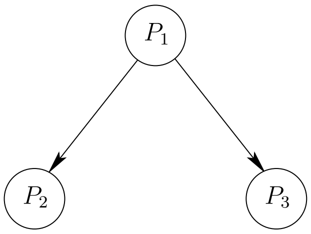
$$\begin{cases} x_2^{1,2} = a, & t_{1,2} \leqslant x_2^{1,2}, \\ x_3^{1,2} + x_3^{1,3} = a, & t_{1,2} \leqslant x_3^{1,2}, \\ x_2^{1,2} = a, & t_{1,3}, \leqslant x_3^{1,3}, \\ x_3^{1,3} + x_3^{2,3} = a, & t_{2,3} \leqslant \frac{1}{2} x_3^{2,3}, \\ x_3^{1,2} = x_3^{2,3}, & \\ & \\ t_1^{out} = t_{1,2} + t_{1,3}, & t_1^{out} \leqslant 1, \\ t_2^{in} = t_{1,2}, & t_2^{in} \leqslant 1, \\ t_2^{out} = t_{2,3}, & t_2^{out} \leqslant 1, \\ t_3^{in} = t_{2,3} + t_{1,3}, & t_3^{in} \leqslant 1 \end{cases}$$

The solution of above linear program is given by

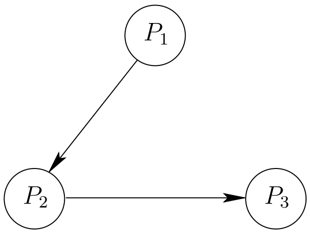
$$x_2^{1,2} = 1, \quad x_3^{1,2} = 1, \quad x_3^{2,3} = 1 \text{ and } a = 1,$$

so that one message can be broadcast every time unit.

Although an optimal schedule reaching this throughput can easily be derived in this small example, this might be a complicated problem in the general case. However, from the solution of above linear program, it is possible (see [4]) to build a set of weighted broadcast trees $(\alpha_1, T_1), \ldots (\alpha_k, T_k)$ such that $\sum_i \alpha_i = a$, thus achieving optimal throughput. The set of trees is determined using graph techniques such as flows and a weighted version of Edmond's branching theorem. Figure 3 depicts the set of all trees that can be used in this decomposition, which consists here in two trees, T_1 and T_2. For our small example, the optimal throughput is achieved using only tree T_2 weighted by 1.



(a) Tree T_1



(b) Tree T_2

Figure 3: Possible broadcast trees of Platform depicted on Figure 2.

The last step of the algorithm consists in determining the actual schedule of the communications induced by above set of trees. Each port (for incoming or outgoing communications) is associated to a node in a weighted bipartite graph. More precisely, we denote by P_j^{out} and P_j^{in} the outgoing and incoming communication ports of P_j and the edge between P_j^{out} and P_k^{in} is weighted by the communication time between

P_j and P_k induced by the set of trees. On our small example, the corresponding bipartite graph is depicted in Figure 4. In order to determine the effective schedule of the communications, a weighted version of König's bipartite graph edge coloring algorithm is used. Clearly, all the communications corresponding to a given matching can be performed simultaneously, and it is proved in [4] that the corresponding schedule of communication achieves optimal throughput.

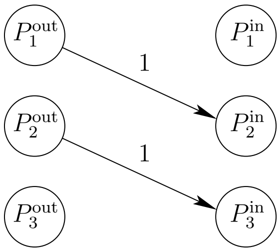


Figure 4: Bipartite graph of communications for the platform depicted on Figure 2 in the bidirectional one-port model.

2.2.2 Unidirectional one-port model

In the case of the unidirectional one-port model, one may think at first sight that the approach depicted in the above section for the bidirectional one-port model could be adapted. However, let us consider again the platform depicted in Figure 2. An equivalent linear program, analogous to the bidirectional case would simply be obtained by changing the constraints corresponding to the occupation of communication ports. In the unidirectional one-port model, the occupation time t_j of the communication port of P_j is simply obtained by summing up the occupation time for both incoming and outgoing communications. For our example, the only change is about node P_2, which is the only node that may send and receives messages. The occupation of this node is now $t_2 = t_{1,2} + t_{2,3}$. This leads to the following linear program:

Unidirectional Steady-State Broadcast Problem on G(Wrong Solution)
Maximize a,
subject to

$$\left\{\begin{array}{ll} x_2^{1,2} = a, & t_{1,2} \leqslant x_2^{1,2}, \\ x_3^{1,2} + x_3^{1,3} = a, & t_{1,2} \leqslant x_3^{1,2}, \\ x_2^{1,2} = a, & t_{1,3} \leqslant x_3^{1,3}, \\ x_3^{1,3} + x_3^{2,3} = a, & t_{2,3} \leqslant \frac{1}{2} x_3^{2,3}, \\ x_3^{1,2} = x_3^{2,3}, & \\ & \\ t_1 = t_{1,2} + t_{1,3}, & t_1 \leqslant 1, \\ t_2 = t_{1,2} + t_{2,3}, & t_2 \leqslant 1, \\ t_3 = t_{2,3} + t_{1,3}, & t_3 \leqslant 1 \end{array}\right.$$

The optimal throughput of the above linear program is $\frac{3}{4}$. The solution can be divided into the two following weighted trees: $(\frac{1}{4}, T_1)$ and $(\frac{1}{2}, T_2)$. Using this set of weighted trees, the communication ports of P_1, P_2 and P_3 are busy during respectively 1, 1 and $\frac{1}{2}$ time units, so that all port constraints are fulfilled. Nevertheless, it is impossible to broadcast $\frac{3}{4}$ messages every time unit, and therefore to achieve the announced throughput. Indeed, in the solution provided by the linear program, the occupation time of the different links is as depicted in Figure 5.

Unfortunately, during the communication between P_1 and P_2, that lasts for $\frac{3}{4}$ time units, no other communication can take place, since the ports of P_1 and P_2 are busy, and thus no communication to P_3 is possible. Then, the communications to P_3 have to be scheduled (the communication from P_2 to P_3 takes $\frac{1}{4}$ time units and the communication from P_1 to P_3 takes $\frac{1}{4}$ time units). Again, those communications have to be sequentialized since both involve the communication port of P_3. Thus, the overall communication time necessary to perform all the communications necessary to send a message of size $\frac{1}{4}$ according to T_1 and a

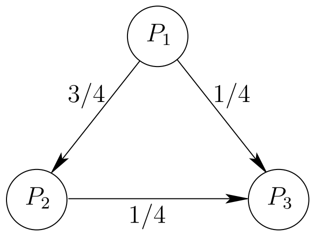
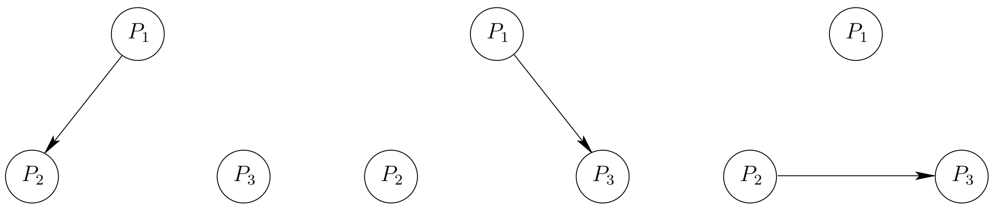


Figure 5: Occupation time of the links in the solution of the first linear program for unidirectional one-port model.

message of size $\frac{1}{2}$ is equal to $\frac{5}{4}$ time units, whereas none of the communication ports is busy during more than one time unit.

In fact, as we will prove it at the end of this section, the set of weighted trees provided using Edmond's branching theorem is not optimal. The optimal solution under the unidirectional one-port model is obtained when sending a message of size $\frac{2}{3}$ according to broadcast tree T_1. In this case the value of the objective function of the linear program is smaller ($\frac{2}{3}$ instead of $\frac{3}{4}$) but the actual throughput achieved in nevertheless larger ($\frac{2}{3}$ instead of $\frac{3}{5}$).

Therefore, in the case of the unidirectional one-port model, we cannot consider separately the construction of the broadcast scheme (i.e. how to determine the set of weighted trees used to perform the broadcast) and the construction of the actual schedule (i.e. how to find the set of matchings used to organize communications). In the case of the bidirectional one-port model, scheduling the communications is equivalent to edge coloring a weighted bipartite graph, which can be solved using a weighted version of König's algorithm so that the total execution time of the schedule corresponds to the maximal occupation time of an incoming or outgoing communication port. As we have seen above, this property does not hold true for the unidirectional one-port model. Indeed, edge coloring general graphs is much more complex than edge coloring weighted bipartite graphs. In particular, even if fractional edge coloring can be solved in polynomial time for general graphs [23, 24], this is not true that the overall sum of the weighted matchings involved in the description is equal to the maximal weighted degree of a vertex.



(a) Matching M_1 (b) Matching M_2 (c) Matching M_3

Figure 6: Possible matchings for platform depicted in Figure 2 in the unidirectional one-port model

Thus, in order solve the broadcast throughput problem under the unidirectional one-port model, we need to simultaneously consider the problems of determining the set of weighted trees used to broadcast the message and the set of weighted matchings used to schedule the communications. Let us again consider the small example depicted in Figure 2. There are 3 possible matchings M_1, M_2 and M_3 (with respective weight x_1, x_2 and x_3) and 2 different possible broadcast trees T_1 and T_2 (with respective weights y_1 and y_2), as depicted in Figures 3 and 6. In order to determine the maximal number of messages that can be broadcast during one time unit, we aim at finding a weighted set of trees of maximal weight $(y_1 + y_2)$ and a set of weighted matchings achieving the communications corresponding to the weighted set of trees, whose overall weight $x_1 + x_2 + x_3$ is smaller than 1. We add a constraint to ensure that on a given link, the set of matchings covers all the communications induced by all broadcast trees. For example, on link $P_1 \rightarrow P_2$, the matchings including this link (here only M_1) has the same weight than the sum of the trees going through this links (here trees T_1 and T_2) multiplied by the time necessary to transfer one message through this link.

This leads to the following linear program

$$
\begin{array}{l}
\textsc{Unidirectional Steady-State Broadcast Problem on G}\\
\textsc{Maximize } y_1 + y_2,\\
\textsc{subject to}\\
\left\{\begin{array}{l}
x_1, \quad x_2, \quad x_3, \quad y_1, \quad y_2 \geqslant 0\\
x_1 + x_2 + x_3 \leqslant 1\\
x_1 = (y_1 + y_2) \times c_{1,2} = y_1 + y_2\\
x_2 = y_1\\
x_3 = y_2 \times c_{1,3} = \frac{1}{2} y_2
\end{array}\right.
\end{array}
$$

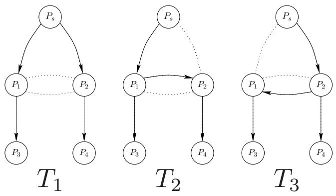
whose solution is given by $y_1 = 0, \quad y_2 = \frac{2}{3}, \quad x_1 = \frac{2}{3}, \quad x_2 = 0, \quad x_3 = \frac{1}{3}$ and therefore $y_1 + y_2 = \frac{2}{3}$. Thus, with respect to the broadcast throughput problem under the unidirectional one-port method, the optimal solution has throughput $\frac{2}{3}$ and consist in sending the messages along tree T_1, as said above.

Therefore, it is possible to find the optimal set of weighted trees and the corresponding set of matchings by solving a linear program. Unfortunately, if this method can be applied to the small platform depicted in Figure 2, both the number of possible matchings and the number of possible broadcast trees may be exponential in the size of the platform graph!

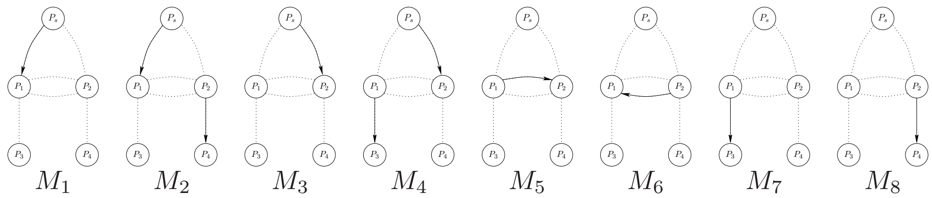
The main contribution of the paper consists in the efficient (polynomial time) resolution of this linear program, as developed in Section 3. However, before going into details, we illustrate why using the whole platform instead of a single broadcast tree in useful to our goal, by comparing the throughput obtained when we allow different network topologies to broadcast a series of messages.

2.3 Comparing topologies for series of broadcasts

In this section, we work out a small example, whose objective is to show the difficulty of the problem. We compare the best throughput that can be achieved using a tree, a directed acyclic graph (DAG), or the full topology with cycles and we prove on the platform depicted in Figure 1, the throughput that can be achieved using the whole platform is strictly larger than the throughput that can be achieved using a DAG only, which is itself strictly larger than the throughput that can be achieved using a single tree. In the platform G depicted Figure 1, there are 3 possible broadcast trees:



and 8 possible matchings:

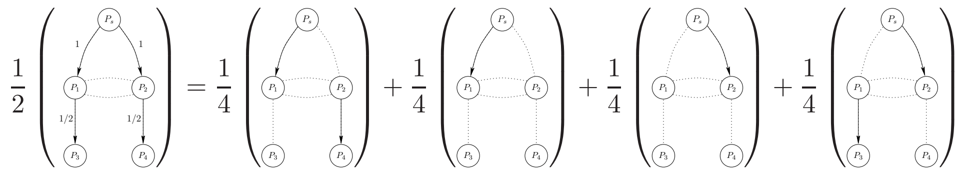


In what follows, we provide for each kind of platform (tree, DAG, general graph) both the set of trees used to broadcast the messages and the set of matchings used to schedule the communications. Our aim is not to describe how the trees and the matchings have been computed. Since the platform is small, the set of weighted trees and weighted matchings have been determined using the linear program described in the previous section.

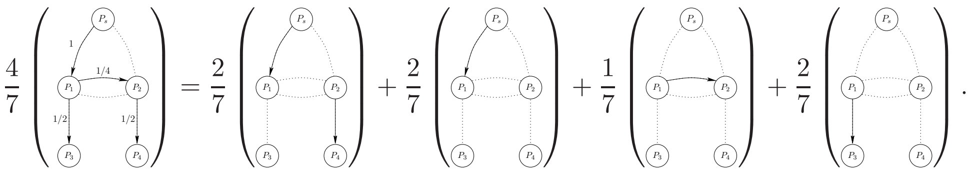
2.3.1 Best tree

We first consider the problem of determining the best tree, subgraph of G. The only possible trees are T_1, T_2 and T_3. Due to symmetry, T_2 and T_3 will achieve exactly the same throughput. Our aim is therefore to determine the maximal number of messages that can be broadcast in one time unit along T_1 and T_2.

Using the linear program introduced in previous section, the throughput that can be achieve using T_1 is $\frac{1}{2}$ and the throughput that can be achieved using T_2 is $\frac{4}{7}$, thus slightly better. The corresponding matchings used to schedule the communications are as follows

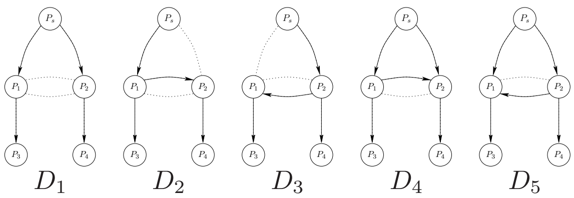


and



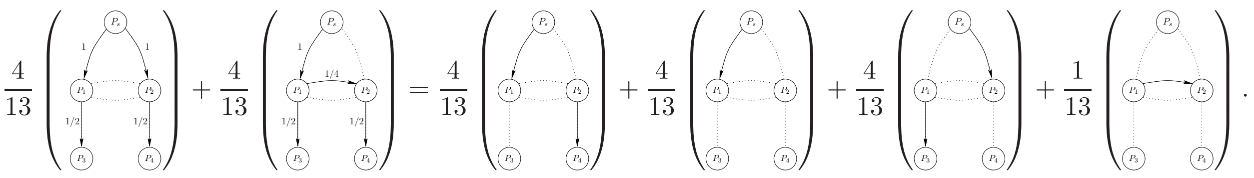
2.3.2 Best DAG

We now consider the problem of determining the best DAG, subgraph of G. There are only 5 different DAGs subgraph of G and such that the messages sent from P_s can reach all the nodes, i.e.



The cases of the first three DAGs (which are trees) have already been addressed and for symmetry reasons, the last two DAGs will lead to the same throughput.

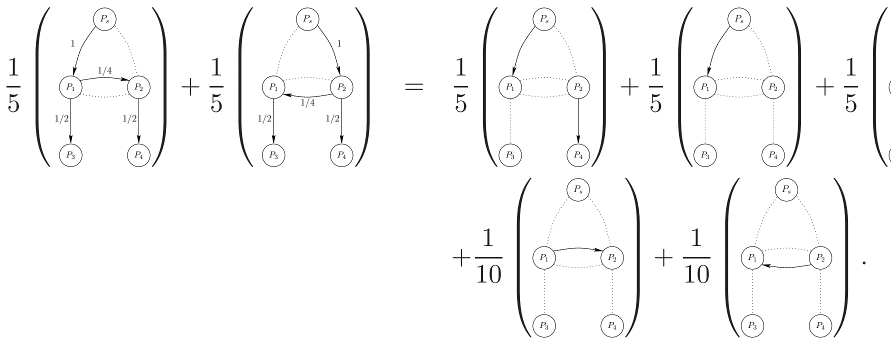
The best throughput that can be achieved using D_1 is $\frac{8}{13}$, thus slightly better than what can be achieved with a single tree. More precisely, during one time unit, $\frac{4}{13}$ of messages can be sent along T_1 and $\frac{4}{13}$ of messages can be sent along T_2. The corresponding matchings used to schedule the communications are as follows



2.3.3 Optimal solution

Let us now consider the case of the whole platform. The best throughput that can be achieved using the whole platform is $\frac{2}{5}$, thus slightly better than what can be achieved with a single tree or even a DAG. More precisely, during one time unit, $\frac{1}{5}$ of messages can be sent along T_2 and $\frac{1}{5}$ of messages can be sent along T_3.

The corresponding matchings used to schedule the communications are as follows



3 Steady-state throughput of the broadcast

In this section, we move to the formal definition of the Series of Broadcast problem in the unidirectional one-port model (which we be called the Broadcast problem in the following for short), and we develop its resolution in the general case. A small example was presented Section 2.2.2, leading to a linear program. In this simple case, it was possible to solve this program since it involves very few variables and constraints. However, in the general case developed in Section 3.2, this formulation may lead in an exponential linear program. The rest of this Section (parts 3.3 to 3.5) is devoted to the description a complicated polynomial algorithm which solves this linear program in polynomial time and provides a polynomial description of an optimal schedule for the Broadcast problem.

3.1 Definition of the Broadcast problem

In the Broadcast operation, we consider a platform represented by an edge-weighted directed graph $G = (V, E, c)$ where $V = (P_1, \ldots, P_n)$ is the set of processors, E is the set of communication links between these processors, and $c(e)$ for $e \in E$ is the *cost of edge* e, that is the time needed to transfer a message of unit size through edge e. We denote by P_{source} the processor initiating the broadcast operation. All other processors have to receive the messages broadcast by the source.

As we focus on the steady-state performance, we assume that P_{source} has a huge number of messages that have to be sent to other processors. We call *throughput* of a schedule for the broadcast operation the average number of messages received by all the processors during one time-unit.

Definition 1 (BROADCAST(G, P_{source})). *Given a platform graph G and a source node P_{source}, find a schedule for a broadcast operation on G from P_{source} achieving the best throughput in the steady-state operation.*

Note that this problem does not belong to NP, as it might be too long to check if a given schedule is valid. To build valid schedule, we have to ensure that its description is polynomial is the size of the initial data. We now introduce a few notations for the broadcast problem.

$\mathcal{T}$ denotes the set of broadcast trees: all spanning trees rooted in P_{source} in G can be used to broadcast one of the messages emitted by P_{source}, so $\mathcal{T}$ is the set of these trees.

$\mathcal{M}$ denotes the set of subset of links that can be used to perform simultaneous communications: if two edges $e_1 = (P_1 \to Q_1)$ and $e_2 = (P_2 \to Q_2)$ belongs to the same subset, this means that P_1 may send a message to Q_1 while P_2 sends a message to Q_2. As we consider here the unidirectional one-port model, each processor may perform only one communication (sending or receiving a message) at a given time step. So $\mathcal{M}$ is the set of the matchings in the platform graph $G = (V, E)$.

Note that these sets $\mathcal{T}$ and $\mathcal{M}$ may not have a polynomial description. To ensure that the solutions built by the algorithm have a polynomial size, we need to describe them with a polynomial number of trees and matchings.

3.2 Constraints and linear program

In this section, we introduce notations so as to express the Broadcast problem as a linear program.

Let us note $\mathcal{T} = \{T_1, \dots, T_{|T|}\}$ and $\mathcal{M} = \{M_1, \dots, M_{|\mathcal{M}|}\}$. The variables of the linear program are the followings:

- x_m is the weight of matching M_m in a solution, that is the average time this matching is used to perform communications during one time-unit.
- y_t is the weight of tree T_t in a solution, that is the average number of message carried by this tree during one time-unit.

The first constraints express that these variables have to be non-negative:

$$\forall M_m \in \mathcal{M}, \quad x_m \geqslant 0, \qquad \forall T_t \in \mathcal{T}, \quad y_t \geqslant 0$$

Then come the constraints expressing that the average communicating can be orchestrated during one time-unit. A matching represents a set of communications that can be scheduled concurrently, but the communications involved in two distinct matchings have to be sequentialized. Thus the total weight of all matchings must be smaller than one-time unit:

$$(comm) \qquad \sum_{M_m \in \mathcal{M}} x_m \leqslant 1$$

The last set of constraints express that the matchings can actually perform all the communications needed by the broadcast trees. On each link, the weight of all matchings including this link (that is the total communication time on this link) must be equal to the total number of messages carried by each tree using this link multiplied by the time necessary to transfer one message:

$$(edge_k) \quad \forall e_k = (i,j) \in E \qquad \sum_{\substack{M_m \in \mathcal{M} \\ (i,j) \in M_m}} x_m = \sum_{\substack{T_t \in \mathcal{T} \\ (i,j) \in T_t}} c_{i,j} \cdot y_t$$

Finally, the objective function is the total weight of the broadcast trees:

$$\max \sum_{T_t \in \mathcal{T}} y_t$$

We get the following linear program :

$$(P) \left\{ \begin{array}{l} \textsc{Maximize} \ \displaystyle\sum_{T_t \in \mathcal{T}} y_t, \\ \textsc{subject to} \\ \qquad\qquad \forall M_m \in \mathcal{M}, \forall T_t \in \mathcal{T}, \quad x_m \geqslant 0, \quad y_t \geqslant 0 \\ (comm) \qquad\qquad \displaystyle\sum_{M_m \in \mathcal{M}} x_m \leqslant 1 \\ (edge_k) \qquad \forall e_k = (i,j) \in E \quad \displaystyle\sum_{\substack{M_m \in \mathcal{M} \\ (i,j) \in M_m}} x_m = \sum_{\substack{T_t \in \mathcal{T} \\ (i,j) \in T_t}} c_{i,j} \cdot y_t \end{array} \right.$$

To solve the Broadcast(G, P_{source}) problem, it is enough to find an optimal solution of (P) consisting in a polynomial number of trees and matchings with non zero weight. In the following, we aim at finding such a solution to (P).

The linear program (P) has a number of constraints potentially exponential in the size of the graph (equals to the number of matchings and trees in G), but has a polynomial number of constraints (equals to $|E| + 1$). The method we want to use to find an optimal solution can be applied on a system with a limited (polynomial) number of variables, but without any bound on the size of the constraints set, as soon as we know some properties on this set, for example if we can decide in polynomial time if a solution satisfy every constraints or not. That is the reason why we compute the dual linear program of (P), denoted by D: (D) has a polynomial number of variables and a potentially exponential number of constraints, but all these constraints are linked to matchings or trees in G, which provides some interesting properties, as developed in Section 3.5.

3.3 Dual linear program for Broadcast

In this section, we compute the dual linear program of (P). We first rewrite the initial linear program:

- variables $X = \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{R}^{|\mathcal{M}|+|\mathcal{T}|}$
- the objective function is Maximize $d^T \cdot X$ where $d = (\underbrace{0, \cdots, 0}_{|M|}, \underbrace{1, \cdots, 1}_{|T|})^T$
- constraints for the one-port model: $A \cdot X \leqslant a$ where $a = 1$ and

$$\begin{cases} \forall M_m \in \mathcal{M}, & A_{1,m} = 1 \\ \forall T_t \in \mathcal{T}, & A_{1,|\mathcal{M}|+t} = 0 \end{cases}$$

- constraints for the matching realizing all the communications needed by the trees: $D \cdot y = b$ where $b = 0 \in \mathbb{R}^{|E|}$ and

$$\begin{cases} \forall k = 1, \ldots, |E| \quad \forall M_m \in \mathcal{M}, \quad D_{k,m} = \begin{cases} -1 & \text{if } e_k \in \mathcal{M}_m \\ 0 & \text{otherwise} \end{cases} \\ \\ \forall k = 1, \ldots, |E| \quad \forall T_t \in \mathcal{T}, \quad D_{k,|\mathcal{M}|+t} = \begin{cases} c_{i,j} & \text{if } e_k = (i,j) \in \mathcal{T}_t \\ 0 & \text{otherwise} \end{cases} \end{cases}$$

The initial linear program is equivalent to the following:

$$\begin{array}{l} \text{Maximize } d^T \cdot X \\ \text{subject to} \\ \quad X \geqslant 0 \\ \quad A \cdot X \leqslant a \\ \quad D \cdot X = b \end{array}$$

These are the same notations as the general formulation of the dual problem [25], with the other matrices (B, C, E, F, G and H) empty. This is equivalent to solve the dual program:

$$\begin{array}{l} \text{Minimize } a \cdot u + b \cdot v \\ \text{subject to} \\ \quad u \geqslant 0 \\ \quad A^T \cdot u + D^T \cdot v \geqslant d \end{array}$$

For m in $\{1, \ldots, |M|\}$, the m^{th} line of the last inequalities gives the constraints corresponding to matching M_m:

$$\begin{aligned} & \left[A^T \cdot u + D^T \cdot v\right]_m \geqslant d_m \\ \Leftrightarrow \quad & A_{1,m} \cdot u + \sum_{k=1}^{|E|} D_{k,m} \cdot v_k \geqslant 0 \\ \Leftrightarrow \quad & u + \sum_{\substack{k=1\ldots|E| \\ e_k \in M_m}} -v_k \geqslant 0 \\ \Leftrightarrow \quad & \sum_{e_k \in M_m} v_k \leqslant u \end{aligned}$$

For t in $\{1, \dots, |T|\}$, the $(|\mathcal{M}| + t)^{\text{th}}$ line of the inequalities gives the constraints corresponding to tree T_t:

$$\begin{aligned} & \left[A^T \cdot u + D^T \cdot v\right]_{|\mathcal{M}|+t} \geqslant d_{|\mathcal{M}|+t} \\ \Leftrightarrow \quad & A_{1,|\mathcal{M}|+t} \cdot u + \sum_{k=1}^{|E|} D_{k,|\mathcal{M}|+t} \cdot v_k \geqslant 1 \\ \Leftrightarrow \quad & \sum_{e_k=(i,j)\in T_t} c_{i,j} \cdot v_k \geqslant 1 \end{aligned}$$

In these constraints, u_i is the variable of the dual linear program corresponding to the initial constraint *comm*, and v_k corresponds to the initial constraint $edge_k$. The dual linear program is summarized by:

$$(D) \begin{cases} \text{MINIMIZE } u, & \\ \text{SUBJECT TO} & \\ & u \geqslant 0 \\ \forall M_m \in \mathcal{M}, & \sum_{e_k \in M_m} v_k \leqslant u \\ \forall T_t \in \mathcal{T} & \sum_{e_k=(i,j)\in T_t} c_{i,j} \cdot v_k \geqslant 1 \end{cases}$$

3.4 Combinatorial optimization problems and results

In this section, we present the formal definition of the separation and optimization problem, and we recall some theoretical results about combinatorial optimization presented in [9]. Let us first introduce the SOPT and SSEP problems:

Definition 2 (The Strong Optimization Problem (SOPT(K, C))). *Given a convex K and a vector $C \in \mathbb{R}^n$, find a vector $x \in K$ that maximizes $C^T \cdot x$ or assert that K is empty.*

Definition 3 (The Strong Separation Problem (SSEP(K, x))). *Given a vector $x \in \mathbb{R}^n$, decided whether $x \in K$, and if not, find a vector hyperplane that separates x from K; more exactly, find a vector $C \in \mathbb{R}^n$ such that $C^T \cdot x > \max\left\{C^T \cdot y \mid y \in K\right\}$.*

The optimization problem of the linear program (D) is equivalent to solve the strong optimization problem $\text{SOPT}(K, C)$ for some K and C. Note that in the following, n denotes the dimension of the solution space: $n = 1 + |E|$. Considering a vector x of this space $\mathbb{R}^n$, the first coordinates correspond to the u variable and the following $|E|$ coordinates are the v_i. Formally, we have: $x_1 = u$ and $x_{k+1} = v_k$ for $k = 1, \dots, |E|$. We may know define formally K and C corresponding to (D):

$$K = \left\{ x \in \mathbb{Q}^n, \quad \begin{array}{ll} & x_1 \geqslant 0 \\ \forall m = 1, \dots, |\mathcal{M}| & \sum_{e_k \in M_m} x_{k+1} \leqslant x_1 \\ \forall t = 1, \dots, |\mathcal{T}| & \sum_{e_k=(i,j)\in T_t} c_{i,j} \cdot x_{k+1} \geqslant 1 \end{array} \right\}$$

and

$$C_1 = -1 \qquad \forall i = 1, \dots, |E| \; C_{i+1} = 0$$

The equivalence between the optimization and the separation problem can be found in [9, chapter 6], and is formulated in the following theorem:

Theorem 1 (Theorem 6.4.9 in [9]). *Any one of the following three problems:*

- *strong separation,*
- *strong violation,*
- *strong optimization,*

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

There remains to define what is a "well-described" polyhedron, and to check that K is well-described. The following definition can be found in [9]:

Definition 4. *Let $P \subseteq \mathbb{R}^n$ be a polyhedron and let φ be a positive integer.*

*(i) We say that (P) has **facet-complexity** at most φ if there exists a system of inequalities with rational coefficients that has solution set (P) and such that the encoding length of each inequality of the system is at most φ.*

*(ii) A **well-described polyhedron** is a triple $(P; n, \varphi)$ where $P \subseteq \mathbb{R}^n$ is a polyhedron with facet-complexity at most φ. The **encoding length** $\langle P \rangle$ of a well-described polyhedron $(P; n, \varphi)$ is $\varphi + n$.*

The polyhedron K corresponding to the optimization problems (D) is clearly expressed as a set of inequalities. Each of these inequalities consists in a sum of at most $|E|$ terms with coefficients either 1 or one of the $c_{i,j}$. So each of these inequalities describing K can be encoded in length polynomial in the size of the platform graph G. Thus, the encoding length for K is polynomial in the size of G. In this special case, the previous theorem leads to the following result:

Lemma 1. *The following properties are equivalent:*

(i) there exists a polynomial-time algorithm that solves the strong optimization problem SOPT(K, c),

(ii) there exists a polynomial-time algorithm that solves the strong separation problem SSEP(K).

In the case of the Broadcast problem, the the separation problem corresponding to the dual linear program can be expressed as follows:

Definition 5 (Broadcast-SSEP(G,$P_{\mathbf{source}}$,x)). *Given a platform graph for the broadcast G, a source node P_{source}, and a vector $x \in \mathbb{R}^{|E|+1}$, is there a vector $C \in \mathbb{R}^n$ such that $C^T \cdot x > \max \left\{ C^T \cdot y, y \in K \right\}$.*

In the next section, we show that the strong separation problem for the linear program (D) is equivalent to a classical graph problem, which can be solved in polynomial time. However, solving the linear program (D) is not equivalent to solving the Compact-Weighted-Broadcast problem: this latter problem is equivalent to solving the linear program (P), as noticed before. Although the value of the objective function is the same in both linear program, we cannot *a priori* derive a solution of (P) from a solution of (D). Fortunately another result in [9] allows us to go from a separation oracle for the dual problem (D) to a an optimal solution to the primal problem (P).

Theorem 2 (Theorem 6.5.15 in [9]). *There exists an oracle-polynomial time algorithm that, for any $c \in \mathbb{Q}^n$ and for any well-described polyhedron $(P; n, \phi)$ 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.

The *basic optimum dual solution with oracle inequalities* is an optimal solution in the dual of the problem defined by the set of inequalities given by the oracle separation. In our case, the separation oracle outputs inequalities of the linear program (D). We call (D') the linear program made up with the same objective function as in (D) and the subset of inequalities answered by the separation oracle. Note that the number of inequalities in (D') is polynomial since the execution of the ellipsoid method involves a polynomial number of calls to the separation oracle. The execution of the ellipsoid method on (D) is exactly the same as the execution on (D). Thus, the maximum objective value α is the same on (D) and on (D'). To simplify the computation of the dual linear program of (D') we add the constraints $u \geqslant 0$ to the set of inequalities in (D'). By adding this constraint, we do not modify the execution of the ellipsoid algorithm on (D'). We call m the total number of constraints obtained. As the execution of the ellipsoid algorithm outputs a polynomial number of constraints, m is polynomial in the size of the initial data.

We now compute the dual of the linear program (D'), which is denoted by (P'). The optimization problem (D') can be written as:

$$\begin{array}{l} \textsc{Minimize } a \cdot u + b \cdot v \\ \textsc{subject to} \\ \quad u \geqslant 0 \\ \quad {A^*}^T \cdot u + {D^*}^T \cdot v \geqslant d^* \end{array}$$

where $A*$ and D^* are selected columns from matrices A and B defined in Section 3.3, corresponding to constraints output by the separation oracle. Using the same formulation of duality, the dual problem of (D') is the following:

$$(P') \left\{ \begin{array}{l} \textsc{Maximize } {d^*}^T \cdot X \\ \textsc{subject to} \\ \quad X \geqslant 0 \\ \quad A^* \cdot X \leqslant a \\ \quad D^* \cdot X = b \end{array} \right.$$

We notice that (P') is equivalent to the linear optimization problem (P) where some variables are set to 0; these variables correspond to inequalities in (D) which have not been selected in (D'). If we denote by $\mathcal{M}^*$ the set of matchings and by $\mathcal{T}^*$ the set of trees corresponding to the inequalities of (D') (and thus, to the variables of (P')), we can rewrite (P') as:

$$(P) \left\{ \begin{array}{l} \textsc{Maximize } \sum_{T_t \in \mathcal{T}} y_t, \\ \textsc{subject to} \\ \quad \forall M_m \in \mathcal{M}^*, \forall T_t \in \mathcal{T}^*, \quad x_m \geqslant 0, \quad y_t \geqslant 0 \\ \qquad\qquad\qquad\qquad \sum_{M_m \in \mathcal{M}^*} x_m \leqslant 1 \\ \qquad \forall e_k = (i,j) \in E \quad \sum_{M_m \in \mathcal{M}^*,\ (i,j) \in M_m} x_m = \sum_{T_t \in \mathcal{T}^*,\ (i,j) \in T_t} c_{i,j} \cdot y_t \end{array} \right.$$

In (P'), the number of variables m and the number of constraints $m + 1 + |E|$ is polynomial in the size of the initial data, so it can be solved in polynomial time. Consider an optimal $(x^{\text{opt}}, y^{\text{opt}})$ solution of (P'). As (P') is the dual problem of (D'), the optimal solution $(x^{\text{opt}}, y^{\text{opt}})$ reaches the same value α of the objective function. This solution provides a polynomial number of trees and matchings with optimal throughput α: this is a compact description of an optimal solution for the broadcast problem. This result is summarized in the following lemma:

Lemma 2. *Given a platform graph for the broadcast G and a source node P_{source}, and a strong separation oracle for the convex defined by the linear inequalities of the corresponding dual problem (D), there exists a polynomial time algorithm that finds an optimal solution of the broadcast problem which consists in a polynomial number of trees and matchings*

To use this result, we have to build a separation oracle for (D), that is to prove that the separation problem in the dual can be solved in polynomial time.

3.5 Separation problem in the dual and spanning tree

In this section, we show that the separation problem for (D) can be solved in polynomial time.

Lemma 3. *Broadcast-SSEP(G,P_{source},x) can be solved in polynomial time.*

Proof. Consider an instance of the strong separation problem on K, that is a vector $x \in \mathbb{R}^n$. We have to choose whether x is in K, and if not, find a hyperplane separating x from K, that is a vector c such that $c^T \cdot x > \max\left\{c^T \cdot y \mid y \in K\right\}$. To answer to this question, it is enough to check if all the inequalities in the definition of K are satisfied for x, and if not, output one inequality which is not satisfied. Once the inequality $u \leqslant 0$ is verified, there remains two types of inequalities defining K: the first inequalities correspond to the matching in $\mathcal{M}$, and the second ones deal with the broadcast trees in $\mathcal{T}$. We consider each type of inequalities:

- Inequalities of type I:
$$\forall m = 1, \ldots, |\mathcal{M}| \sum_{e_k \in M_m} x_{k+1} \leqslant u$$
Consider the directed edge-weighted graph $G_{\mathcal{M}}$ where the vertices and the edges are the same as the platform graph G, and the weight of an edge $e_k = (i,j)$ is $c_{\mathcal{M}}(e_k) = x_{k+1}$. There exists a polynomial time algorithm which finds a matching $M_{\min}$ with maximum weight in $G_{\mathcal{M}}$. If the weight $c_{\mathcal{M}}(M_{\min})$ of this matching is less than u, then for every matching $M \in \mathcal{M}$, $c_{\mathcal{M}}(M) \leqslant u$. Otherwise, this gives an inequality which is not satisfied for x:
$$\sum_{e_k \in M_{\min}} x_{k+1} > u$$
To make it short, solving the separation problem for this set of inequalities can be done by searching a maximum weight matching.

- Inequalities of type II:
$$\forall t = 1, \ldots, |\mathcal{T}| \sum_{e_k=(i,j) \in T_t} c_{i,j} \cdot x_{k+1} \geqslant 1$$
where $\mathcal{T}$ is the set of all spanning trees in G. We consider the directed graph $G_{\mathcal{T}}$ consisting in the same set of nodes and edges as G, and with weight of edge $e_k = (i,j)$ equal to $c_{\mathcal{T}}(i,j) = c_{i,j} \cdot x_{k+1}$. There exists a polynomial time algorithm that finds a tree $T_{\min}$ with minimum weight in $G_{\mathcal{T}}$. If the weight of this tree is greater than 1, then all inequalities of type II are satisfied. Otherwise, we have:
$$\sum_{e_k=(i,j) \in T_{\min}} c_{i,j} \cdot x_{k+1} < 1$$
which provides a hyperplane separating x from K. ∎

Using Lemma 2 we can express the following result:

Theorem 3. *Computing the best throughput of a broadcast operation in steady-state can be solved in polynomial time.*

We have to consider these results as theoretical results, with little practical interest since the ellipsoid method used in Theorem 2 may be very time-consuming.

4 Other collective communication schemes

This framework can be applied to other communication schemes for obtain a polynomial algorithm for the unidirectional one-port model. In this section, we extend these results to the Scatter and All-to-all collective communications.

4.1 Data distribution: the Scatter operation

In the *Scatter* operation, one source processor P_{source} has to send a distinct message to each target processor $P_{\text{target}_1}, \ldots P_{\text{target}_N}$. As we consider the steady-state operation, we study the *Series of Scatter* scheme. Thus we suppose that the same source processor has a large number of different messages to send to each target. In the steady-state, we define the throughput of such a scatter operation as the average number of messages receives in each target during one time-unit. We can formally define the Scatter problem as follows:

Definition 6 (Scatter(G,$P_{\mathbf{source}}$,($P_{\mathbf{target}_1}, \ldots, P_{\mathbf{target}_N}$))). *Given a platform graph G, a source processor P_{source}, and a set of target processors $\mathcal{P}_{target} = \{P_{target_1}, \ldots, P_{target_N}\}$, find a schedule for a Scatter operation from P_{source} and targeting $\mathcal{P}_{target}$ achieving the best throughput.*

We have already studied this collective communication scheme in [17] considering the bidirectional one-port model. To study the unidirectional one-port model, we have to adapt the previous framework. The main difference with the broadcast operation is the set of communication patterns $\mathcal{T}$. The elementary pattern that can be use to perform a scatter operation is a set of N paths going from P_{source} to each target P_{t_i}.

Let us denote by $\mathcal{P}_i$ the set of the paths from P_{source} to P_{target_i}. Then the set of elementary communication patterns is $\mathcal{T} = \mathcal{P}_1 \times \mathcal{P}_2 \times \cdots \times \mathcal{P}_N = \{(p_1, p_2, \ldots, p_N),\ p_1 \in \mathcal{P}_1, p_1 \in \mathcal{P}_2, \ldots, p_N \in \mathcal{P}_N\}$. With this new set $\mathcal{T}$, we construct the same linear program (P), whose optimization is equivalent to finding an optimal schedule for the Scatter operation. The dual linear program (D) is the same as previously, but leads to a slightly different separation problem, which is denoted by Scatter-SSEP(G, P_{source},x). However, this separation problem is also polynomial:

Lemma 4. *Scatter-SSEP(G, P_{source},x) can be solved in polynomial time.*

Proof. As previously, we consider a vector x and try either to prove that x belongs to the convex K defined by the inequalities of the dual (D) or to find an hyperplane separating x from K. To this goal, we check if all inequalities in the definition of K are satisfied. We recall that these inequalities are of two types: the first ones correspond to the matching in $\mathcal{M}$, and the second ones deal with the communication patterns in $\mathcal{T}$.

- Inequalities of type I: we can check if all these constraints are verified in polynomial time as previously with the search of a minimum matching in $G_{\mathcal{M}}$.

- Inequalities of type II:
$$\forall t = 1, \ldots, |\mathcal{T}| \qquad \sum_{e_k=(i,j)\in T_t} c_{i,j} \cdot x_{k+1} \geqslant 1$$
Since each scheme t in $\mathcal{T}$ is made up with paths from P_{source} to each target P_{target_i}, we can rewrite these inequalities:
$$\forall (p_1, \ldots, p_N) \in \mathcal{P}_1 \times \cdots \times \mathcal{P}_N \quad \sum_{u=1}^{N} \left(\sum_{e_k=(i,j)\in p_u} c_{i,j} \cdot x_{k+1} \right) \geqslant 1$$
We now consider the directed graph $G_{\mathcal{T}}$ consisting in the nodes and edges of G, with weight of edge $e_k = (i,j)$ set to $c_{\mathcal{T}}(i,j) = c_{i,j} \times x_{k+1}$. In polynomial time, we can find, for each target P_{target_i} a path $p_i^{\min}$ of minimum cost in $G_{\mathcal{T}}$ from P_{source} to P_{target_i}. If the pattern $(P_1^{\min}, \ldots, P_N^{\min})$ has a weight greater than 1, then all other patterns in $\mathcal{T}$ has a weight greater than 1, and all inequalities of type II are satisfied. Otherwise, we have:
$$\sum_{u=1}^{N} \left(\sum_{e_k=(i,j)\in p_u^{\min}} c_{i,j} \cdot x_{k+1} \right) < 1$$
which provides a hyperplane separating x from K. ■

4.2 Data sharing : the Total-Exchange operation

Another collective communication scheme close is an extension of the Broadcast problem and is called *Total-Exchange*: in this scheme, all processors own a data and want to broadcast this data to all other processors. This communication scheme is calld the *"AllGather"* operation in the MPI standard. In the steady-state operation, we suppose that each processor has a set of data to broadcast to other processors. We define the throughput of the Total-Exchange operation as the throughput of all broadcast operation involved in this scheme. We formally define the Total-Exchange problem as follows:

Definition 7 (Total-Exchange(G)). *Given a platform graph G, find a schedule for an Total-Exchange operation achieving the best throughput in the steady state operation.*

As previously, we extend the study of the Broadcast problem with a little change on the set of communication patterns. $\mathcal{T}$ now contains collections of broadcast trees: one tree for each source node in the platform graph. If we denote by $\mathcal{T}_i$ the set of spanning trees rooted at P_i, we have:

$$\mathcal{T} = T_1 \times T_2 \times \cdots \times T_n$$

We can use the framework developed for the Broadcast problem, provided that we can solve the separation problem on the convex defined by the inequalities of the dual linear program. The only with the separation problem for the Broadcast are the inequalities of type II:

$$\forall (t_1, t_2, \ldots, t_n) \in \mathcal{T}_1 \times T_2 \times \cdots \times T_n \quad \sum_{u=1\ldots n} \left(\sum_{e_k=(i,j) \in t_u} c_{i,j} \cdot x_{k+1} \right) \geqslant 1$$

We can build in polynomial time n minimum spanning trees $t_1, \ldots, t_n$ in $G_{\mathcal{T}}$ such that $t_i^{\min}$ is rooted in P_i. If the previous inequality is satisfied for $(t_0^{\min}, \ldots, t_n^{\min})$, then it is verify for each collection of trees in $\mathcal{T}$. Otherwise, this gives an hyperplane separating x from K.

4.3 Personalized data exchange: the All-To-All operation

We apply our framework to a last communication scheme, known as the (personalized) "All-To-All" operation, which is close to the Scatter operation, In this new scheme, several sources $P_{\text{source}1}, \ldots, P_{\text{source}M}$ aim at sending distinct messages to several targets $P_{\text{target}_1}, \ldots, P_{\text{target}_N}$ (the messages are differentiated both by source and target). In the steady-state All-To-All operation, each source is supposed to have a big number of messages to send to each target. The throughput of such an All-To-All operation is the average number of messages receives in each target from each source during one time-unit.We can formally define the All-To-All problem as follows:

Definition 8 (All-To-All(G,($P_{\textbf{source}1}, \ldots, P_{\textbf{source}M}$),($P_{\textbf{target}_1}, \ldots, P_{\textbf{target}_N}$))). *Given a platform graph G, a set of source processors $\mathcal{P}_{source} = \{P_{source1}, \ldots, P_{sourceM}\}$ and a set of target processors $\mathcal{P}_{source} = \{P_{target_1}, \ldots, P_{target_N}\}$, find a schedule for an All-To-All operation from $\mathcal{P}_{source}$ to $\mathcal{P}_{target}$ achieving the best throughput in the steady state operation.*

The previous study about the Scatter operation can be extended to this communication scheme by changing the set $\mathcal{T}$: an elementary communication pattern is now a set of $N \times M$ paths containing a path from each source to each target. If we denote by $\mathcal{P}_{i,j}$ the set of all paths going from P_i to P_j, we have:

$$\mathcal{T} = \prod_{\substack{i=1\ldots M \\ j=1\ldots N}} \mathcal{P}_{\text{source}_i,\text{target}_j} = \begin{array}{c} \mathcal{P}_{\text{source}_1,\text{target}_1} \times \cdots \times \mathcal{P}_{\text{source}_1,\text{target}_N} \\ \times \mathcal{P}_{\text{source}_2,\text{target}_1} \times \cdots \times \mathcal{P}_{\text{source}_2,\text{target}_N} \\ \vdots \\ \times \mathcal{P}_{\text{source}_M,\text{target}_1} \times \cdots \times \mathcal{P}_{\text{source}_M,\text{target}_N} \end{array}$$

Using the proof of the Scatter case, we can build a polynomial time algorithm that computes an optimal solution to the All-To-All problem, provided that we are able to solve the separation one the convex defined by the inequalities of the dual linear program (D). The only difference with the previous separation problem is the inequalities of type II, which are:

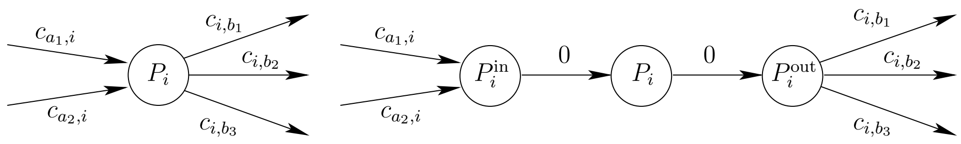
$$\forall (p_{1,1}, \ldots, p_{M,N}) \in \mathcal{T} \quad \sum_{\substack{u=1\ldots M \\ v=1\ldots N}} \left(\sum_{e_k=(i,j) \in p_{u,v}} c_{i,j} \cdot x_{k+1} \right) \geqslant 1$$

Once again, to check if one of these inequalities is not satisfied, we can look for minimum paths from each source to each target in the graph $G_{\mathcal{T}}$, and verify this condition on the pattern made up with all minimum paths. Since this can be done in polynomial time, this solves the separation problem in polynomial time, thus there exists a polynomial time algorithm for the All-To-All problem.

5 Case of a complex platform

In previous sections, we have provided a polynomial time algorithm to find the best schedule for some communications schemes (Broadcast, Scatter, ...) with respect to the throughput under the one-port model. In a complex and large platform, the different nodes may well be equipped with different network interfaces. In particular, some nodes may be able to simultaneously send and receive one message (bidirectional one-port model), and some other nodes may even be able to send and receive several messages at the same time. In this section, we prove that we handle all these cases using only unidirectional one-port nodes. This is simply done by slightly transforming the initial platform.

Let us first consider the case of a node in the bidirectional one-port model, as depicted in Figure 7(a). In order to transform P_i to an equivalent platform where nodes respect unidirectional one-port model, we add two fictitious nodes P_i^{in} and P_i^{out} as depicted in Figure 7(b).

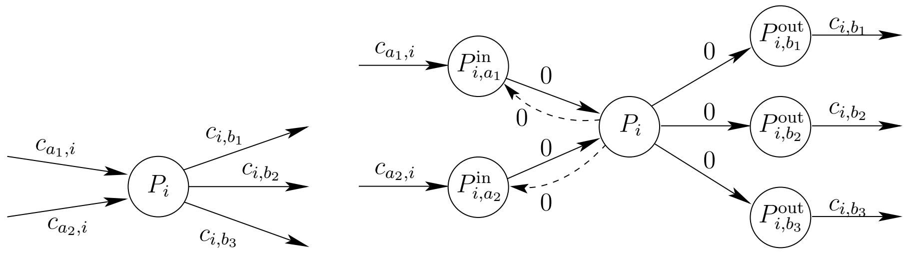


(a) Initial node (b) Node simulated using one-port nodes

Figure 7: Simulating a bidirectional one-port node (on the left) with three unidirectional one-port nodes (on the right).

In the transformed platform, the messages sent to P_i^{in} are sequentialized but P_i^{in} can send messages concurrently to P_i (since $c_{i^{\text{in}},i} = 0$) and the messages sent by P_i^{out} are sequentialized, but P_i^{out} can receive messages concurrently from P_i (since $c_{i,i^{\text{out}}} = 0$).

We can also find an equivalent transformation for nodes under Δ-port model, i.e. that can send and receive several messages concurrently. Again, we can transform any node P_i to an equivalent platform whose nodes respect unidirectional one-port model, by adding fictitious nodes, as depicted on Figure 8.



(a) Initial node (b) Node simulated using one-port nodes

Figure 8: Simulating a Δ-port node with several unidirectional one-port nodes

We also need to add fictitious 0-edges between P_i and $P_{j,i}^{\text{in}}$ in the case of the broadcast, to be sure that every nodes will receive all messages at the end of the broadcast operation: as soon as P_i holds the messages, received from one of the $P_{j,i}^{\text{in}}$ nodes (say $P_{a_1,i}^{\text{in}}$ for example) it can send it to other such nodes (like $P_{a_2,i}^{\text{in}}$) in time 0 using these new edges.

Thus, after performing above transformations, we can find the optimal broadcast scheme using the algorithm depicted in Section 3, for a complex platform consisting in nodes working either an unidirectional one-port model, bidirectional one-port model or Δ-port model.

6 Asymptotic optimality

In this section, we prove that the previous periodic schedule is asymptotically optimal: basically, no scheduling algorithm (even non periodic) can execute more broadcast operations in a given time-frame than ours, up to a constant number of operations. These results are inspired by the work of Bertsimas and Gamarnik [5], who use a fluid relaxation technique to prove the asymptotic optimality of a simpler packet routing problem. This section is devoted to the formal statement of this result, and to the corresponding proof.

Given a platform graph $G = (V, E, c)$, a source processor P_{source} holding an infinite number of unit-size messages, and a time bound K, define $opt(G, K)$ as the optimal number of messages that can be received by every target processor in a succession of broadcast operations from the source processor, within K time-units. Let $\mathrm{TP}(G) = \sum y_t$ be the solution of the linear program (P) of Section 3 applied to this platform graph G. We have the following result:

Lemma 5. $opt(G, K) \leqslant \mathrm{TP}(G) \times K$

Proof. Consider an optimal schedule, such that the number of messages broadcast by the source processor within the K time-units is maximal, i.e. is equal to $opt(G, K)$. Each of these messages has followed a given tree to reach all processors. Let n_t the number of such messages broadcast by the source which go through tree T_t. We have $\sum_{T_t \in \mathcal{T}} n_t = opt(G, K)$.

Let s_i, $(i = 1 \dots N)$ denotes the dates in the optimal schedule when communications start or stop, such that $s_i < s_{i+1}$. Thus the considered schedule can be divided into time intervals $[s_i, s_{i+1}]$ such that no communication start or stop during one of these time intervals. As the optimal schedule satisfies the unidirectional one-port model, during one interval, the scheduled communications do not involve a given node in more than one transfer, so these communications define a matching in G. Let $M_{f(i)}$ denotes the matching involved during interval $[s_i, s_{i+1}]$. Let t_i denotes the total time matching M_i is involved during the considered schedule. We have

$$t_i = \sum_{\substack{j=1\dots N-1 \\ f(j)=i}} (s_{j+1} - s_j)$$

Then the following constraints hold true:

- the total time of all interval is less than the duration of the schedule:

  $$\sum_{j=1\dots N-1} (s_{j+1} - s_j) \leqslant opt(G, K)$$

  if we gather the time of intervals by matchings, we get:

  $$\sum_{j=1\dots N-1} s_{j+1} - s_j = \sum_{M_i \in \mathcal{M}} \left(\sum_{\substack{j=1\dots N-1 \\ f(j)=i}} s_{j+1} - s_j \right) = \sum_{M_i \in \mathcal{M}} t_i$$

  which leads to

  $$\sum_{M_i \in \mathcal{M}} t_i \leqslant K$$

- Consider link $P_j \to P_k$. The total number of messages going through this link in the optimal schedule is $\sum_{T_t \in \mathcal{T}_{j,k}} n_t$ where $\mathcal{T}_{j,k}$ is the set of trees which include the edge (j, k). The total occupation time of this link in the schedule is $\sum_{M_i \in \mathcal{M}_{j,k}} t_i$ where $\mathcal{M}_{j,k}$ is the set of matchings which include the edge (j, k). So we have the following equality:

  $$\sum_{\substack{M_i \in \mathcal{M} \\ (j,k) \in M_i}} t_i = c_{j,k} \times \sum_{\substack{T_t \in \mathcal{T} \\ (j,k) \in T_t}} n_t$$

Let $y_t = n_t/K$ and $x_m = t_m/K$. All the constraints of (P) hold for the x_m's and the y_t's, hence $\sum n_t/K \leqslant \mathrm{TP}(G)$ since $\mathrm{TP}(G)$ is the optimal value. So we have $opt(G,K)/K \leqslant \mathrm{TP}(G)$. ∎

Again, this lemma states that no schedule can send more messages that the steady-state. There remains to bound the loss due to the initialization and the clean-up phase in our periodic solution, to come up with a well-defined scheduling algorithm based upon steady-state operation. Consider the following algorithm (assume that K is large enough):

- Solve the linear program (P) for the target platform G using the method developed in Section 3. This gives a set of of weighted trees $\mathcal{T}^*$ and matchings $\mathcal{M}^*$ reaching the optimal throughput $\mathrm{TP}(G)$.
- Let L be the maximum depth of all trees in $\mathcal{T}^*$. Send $\mathrm{TP}(G) \times (K-L)$ messages using the set of trees and matchings.
- Since the maximum depth of the broadcast trees is L, it takes at most L time-units for a message to reach its target nodes. So in K time-units, we can initiate and complete the broadcast of $steady(G,K) = \mathrm{TP}(G) \times (K-L)$ messages.

We have proved the following result:

Theorem 4. *The previous scheduling algorithm based on the steady-state operation is asymptotically optimal:*

$$\lim_{K \to +\infty} \frac{steady(G,K)}{opt(G,K)} = 1.$$

Proof. Using the previous lemma, $opt(G,K) \leqslant \frac{1}{\mathrm{TP}(G)} \times K$. From the description of the algorithm, we have $steady(G,K) = \mathrm{TP}(G) \times (K-L)$, hence the result because L, and $\mathrm{TP}(G)$ are constants independent of K. ∎

7 Related Work

The *atomic broadcast* problem has been studied under different models to deal with the heterogeneity of the target architecture. Banikazemi et al. [1] consider a simple model in which the heterogeneity among processors is characterized by the speed of the sending processors. In this model, the interconnection network is fully connected (a complete graph), and each processor P_i requires t_i time-units to send a (normalized) message to any other processor. The authors discuss that this simple model of heterogeneity can well describe the different communication delays in a heterogeneous cluster. They introduce the Fastest Node First (FNF) heuristic: to construct a good broadcast tree, it is better to put fastest processors (processors that have the smallest sending time) at the top of the tree. Some theoretical results (NP-completeness and approximation algorithms) have been developed for the problem of broadcasting a message in this model: see [10, 20, 19, 14].

A more complex model is introduced in [2]: it takes not only the time needed to send a message into account, but also the time spent for the transfer through the network, and the time needed to receive the message. All these three components have a fixed part, and a part proportional to the length of the message.

Yet another model of communication is introduced in [7, 6]: the time needed to transfer the message between any processor pair (P_i, P_j) is supposed to be divided into a start-up cost $T_{i,j}$ and a part depending on the size m of the message and the transmission rate $B_{i,j}$ between the two processors, $\frac{m}{B_{i,j}}$. Since the message size is a constant in the case of a broadcast, the total communication time between P_i and P_j is $C_{i,j} = T_{i,j} + \frac{m}{B_{i,j}}$. In [7], some heuristics are proposed for the broadcast and the multicast using this model.

All previous models assume the one-port protocol, either bidirectional or unidirectional. Usually, when considering atomic broadcast, these two flavors of the one-port model are equivalent: a node receives the broadcast message only once, and is able to forward it only once the receive operation is done. Thus even if we assume the bidirectional one-port model, there is no overlap between reception and emission, so the schedule also satisfy the unidirectional one-port model.

Other collective communications, such as multicast, scatter, all-to-all, gossiping, and gather (or reduce) have been studied in the context of heterogeneous platforms: see [21, 18] and the references provided in [3].

8 Conclusion

In this paper, we have dealt with the problem of broadcasting a series of messages on heterogeneous platforms using the unidirectional one-port model. Our major objective was to maximize the throughput that can be achieved in steady-state mode, when a large number of same-size broadcasts are performed in a pipeline fashion. Achieving the best throughput may well require that the target platform is used in totality: we have shown neither spanning trees nor DAGs are powerful enough.

We have shown that our previous study using bidirectional one-port model cannot be applied in this case. However, we have shown how to compute the best throughput and an optimal schedule using a complex linear programing tool (the ellipsoid method). The solution is then expressed as a set of broadcast trees weighted by their throughput, and a set of weighted matchings describing how to orchestrate the communications simultaneously with respect to the one-port model. This study has been extended to other collective communication schemes such as Scatter, All-To-All, and Total-Exchange. We have also shown that the unidirectional one-port model can be used to emulate others models (bidirectional one-port model or multi-port model).

It is important to point out that a concrete scheduling algorithm for the broadcast problem based upon the steady-state operation is asymptotically optimal, in the class of all possible schedules (not only periodic solutions).

References

[1] M. Banikazemi, V. Moorthy, and D. K. Panda. Efficient collective communication on heterogeneous networks of workstations. In *Proceedings of the 27th International Conference on Parallel Processing (ICPP'98)*. IEEE Computer Society Press, 1998.

[2] M. Banikazemi, J. Sampathkumar, S. Prabhu, D.K. Panda, and P. Sadayappan. Communication modeling of heterogeneous networks of workstations for performance characterization of collective operations. In *HCW'99, the 8th Heterogeneous Computing Workshop*, pages 125–133. IEEE Computer Society Press, 1999.

[3] O. Beaumont, A. Legrand, L. Marchal, and Y. Robert. Optimizing the steady-state throughput of broadcasts on heterogeneous platforms heterogeneous platforms. Technical report, LIP, ENS Lyon, France, June 2003.

[4] O. Beaumont, A. Legrand, L. Marchal, and Y. Robert. Pipelining broadcasts on heterogeneous platforms. In *International Parallel and Distributed Processing Symposium IPDPS'2004*. IEEE Computer Society Press, 2004.

[5] D. Bertsimas and D. Gamarnik. Asymptotically optimal algorithm for job shop scheduling and packet routing. *Journal of Algorithms*, 33(2):296–318, 1999.

[6] P.B. Bhat, C.S. Raghavendra, and V.K. Prasanna. Adaptive communication algorithms for distributed heterogeneous systems. *Journal of Parallel and Distributed Computing*, 59(2):252–279, 1999.

[7] P.B. Bhat, C.S. Raghavendra, and V.K. Prasanna. Efficient collective communication in distributed heterogeneous systems. In *ICDCS'99 19th International Conference on Distributed Computing Systems*, pages 15–24. IEEE Computer Society Press, 1999.

[8] F. Desprez, P. Fraigniaud, and B. Tourancheau. Successive Broadcast on Hypercube. Technical Report CS-93-210, The University of Tennessee - Knoxville USA, 1993.

[9] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithm and Combinatorial Optimization*. Algorithms and Combinatorics 2. Springer-Verlag, 1994. Second corrected edition.

[10] N.G. Hall, W.-P. Liu, and J.B. Sidney. Scheduling in broadcast networks. *Networks*, 32(14):233–253, 1998.

[11] K. Hwang and Z. Xu. *Scalable Parallel Computing*. McGraw-Hill, 1998.

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

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

[14] Samir Khuller and Yoo-Ah Kim. On broadcasting in heterogenous networks. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1011–1020. Society for Industrial and Applied Mathematics, 2004.

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

[16] V. Kumar, A. Grama, A. Gupta, and G. Karypis. *Introduction to Parallel Computing*. The Benjamin/Cummings Publishing Company, Inc., 1994.

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

[18] R. Libeskind-Hadas, J. R. K. Hartline, P. Boothe, G. Rae, and J. Swisher. On multicast algorithms for heterogeneous networks of workstations. *Journal of Parallel and Distributed Computing*, 61(11):1665–1679, 2001.

[19] P. Liu. Broadcast scheduling optimization for heterogeneous cluster systems. *Journal of Algorithms*, 42(1):135–152, 2002.

[20] P. Liu and T.-H. Sheng. Broadcast scheduling optimization for heterogeneous cluster systems. In *SPAA'2000, 12th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 129–136. ACM Press, 2000.

[21] P. Liu and D.-W. Wang. Reduction optimization in heterogeneous cluster environments. In *14th International Parallel and Distributed Processing Symposium (IPDPS'2000)*. IEEE Computer Society Press, 2000.

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

[23] Ljubomir Perkovic. *Edge Coloring, Polyhedra and Probability*. PhD thesis, Carnegie Mellon University, Pittsburgh, 1998.

[24] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24 of *Algorithms and Combinatorics*. Springer-Verlag, 2003.

[25] Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, New York, 1986.

[26] M. Snir, S. W. Otto, S. Huss-Lederman, D. W. Walker, and J. Dongarra. *MPI the complete reference*. The MIT Press, 1996.

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

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

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