Complexity results and heuristics for pipelined multicast operations on heterogeneous platforms

O. Beaumont
LaBRI, UMR CNRS 5800
Bordeaux, France
Olivier.Beaumont@labri.fr

A. Legrand and L. Marchal and Y. Robert
LIP, UMR CNRS-INRIA 5668
ENS Lyon, France
{Arnaud.Legrand,Loris.Marchal,Yves.Robert}@ens-lyon.fr

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, such as multicast operations, where messages are broadcast to a set of predefined targets. We assume that there is a large number of messages to be multicast in pipeline fashion, and we seek to maximize the throughput of the steady-state operation. We target heterogeneous platforms, modeled by a graph where links have different communication speeds. We show that the problem of computing the best throughput for a multicast operation is NP-hard, whereas the best throughput to broadcast a message to every node in a graph can be computed in polynomial time. Thus, we introduce several heuristics to deal with this problem and prove that some of them are approximation algorithms. We perform simulations to test these heuristics and show that their results are close to a theoretical upper bound on the throughput that we obtain with a linear programming approach.

1. Introduction

Multicasting is a key communication primitive in computer networks. Lin and Ni [7] 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. Several authors have discussed optimized broadcast algorithms for a variety of parallel architectures, such as wormhole routed networks, cut-through routed networks, and networks of workstations. Recently, the design of multicast algorithms has been the focus of many papers, due to the advent of new technologies such as mobile, wireless, ad-hoc, and optical networks.

In this paper, we consider multicast algorithms for heterogeneous networks of workstations. We assume a realistic model of communications, namely the *one-port* model, where a given processor can simultaneously receive data from one of its neighbor, and send (independent) data to one of its neighbor at a given time-step. This is to be contrasted with the traditional *multi-port* model, where the number of simultaneous messages sent or received by a given processor is not bounded.

The traditional objective of multicast algorithms is to minimize the *makespan*, i.e. the time elapsed between the emission of the first message by the source and the last reception. In this paper, rather than concentrating on the implementation of a single multicast operation, we deal with the optimization of *a series of successive multicast operations* or equivalently of a single multicast operation of a very large message split into small chunks. Such series of multicasts typically occur in the execution of a complex application, deployed on a heterogeneous "grid" platform, and using macro-communication schemes intensively. In many cases, the application would perform a large number of instances of multicasts (for example if data parallelism is used), and the makespan is not a significant measure for such problems. Rather, we focus on the optimization of the steady-state mode, and we aim at optimizing the averaged throughput, i.e. the averaged number of multicasts which are initiated every time-step.

In previous papers, we have dealt with other communication primitives than the multicast operation. We have shown how to compute the optimal steady-state throughput for a series of scatter or reduce operations [6], and a series of broadcast operations [2].

The idea is to characterize the steady-state operation of each resource through a linear program in rational numbers (that can thus be solved with a complexity polynomial in the platform size), and then to derive a feasible periodic schedule from the output of the program (and to describe this schedule in polynomial size too). In this paper, we prove that surprisingly, multicast operations turns out to be more difficult than scatters or broadcasts: even characterizing the optimal throughput of a series of multicasts is shown to be NP-hard.

Following this negative result, we introduce several polynomial time heuristics to deal with the series of multicasts problem. These heuristics can be divided into two categories: the first category is based upon the linear programming approach, and some heuristics are in fact shown to be approximation algorithms (they have a guaranteed worst-case performance). The second category re-visits the traditional heuristics that aim at building *"good"* multicast trees, namely trees that minimize either the sum of the edge weights (Steiner trees) or the weight of the longest path in the tree (which is the makespan under the multiport model); we extend these heuristics to cope with the new objective, i.e. maximizing the throughput of the multicast tree. Due to lack of space, we do not survey related work: instead, we refer to the extended version of the paper [1].

2. Framework

The target architectural platform is represented by an edge-weighted directed graph $G = (V, E, c)$, as illustrated in Figure 1(a). Note that this graph may well include cycles and multiple paths. Let $p = |V|$ be the total number of nodes. There is a *source* node P_{source}, which plays a particular role: it is the source of all the messages to be sent; initially, it holds all the data to be multicast. There is a set of N destination nodes, which we denote as $\mathcal{P}_{\text{target}} = \{P_{t_1}, \dots, P_{t_N}\}$. If $\mathcal{P}_{\text{target}} = V \setminus \{P_{\text{source}}\}$, all nodes are receiving the messages, we have a succession of broadcast operations. Otherwise, there are some nodes that are neither source nor destination, but which may participate by forwarding the information.

There are several scenarios for the operation mode of the processors. In this paper, we concentrate on the *one-port model*, where a processor node can simultaneously receive data from one of its neighbor, and send (independent) data to one of its neighbor. At any given time-step, there are at most two communications involving a given processor, one in emission and the other in reception.

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 one unit-size message from P_j to P_k. The graph is directed: the time to communicate in the reverse direction, from P_k to P_j, is $c_{k,j}$ (provided that this link exists). 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 before time-step $t+c_{j,k}$ (but it can perform a send operation); and (ii) P_j cannot initiate another send operation before time-step $t + c_{j,k}$ (but it can perform a receive operation).

Series of multicasts We define the Series of Multicasts problem as follows: the source processor emits 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, which we denote as $\textsc{Series}(V, E, c, P_{\text{source}}, \mathcal{P}_{\text{target}})$, is to maximize the throughput, i.e. the average number of multicasts initiated per time-unit. We work out a little example in Section 3, using the platform represented in Figure 1(a).

3. Example

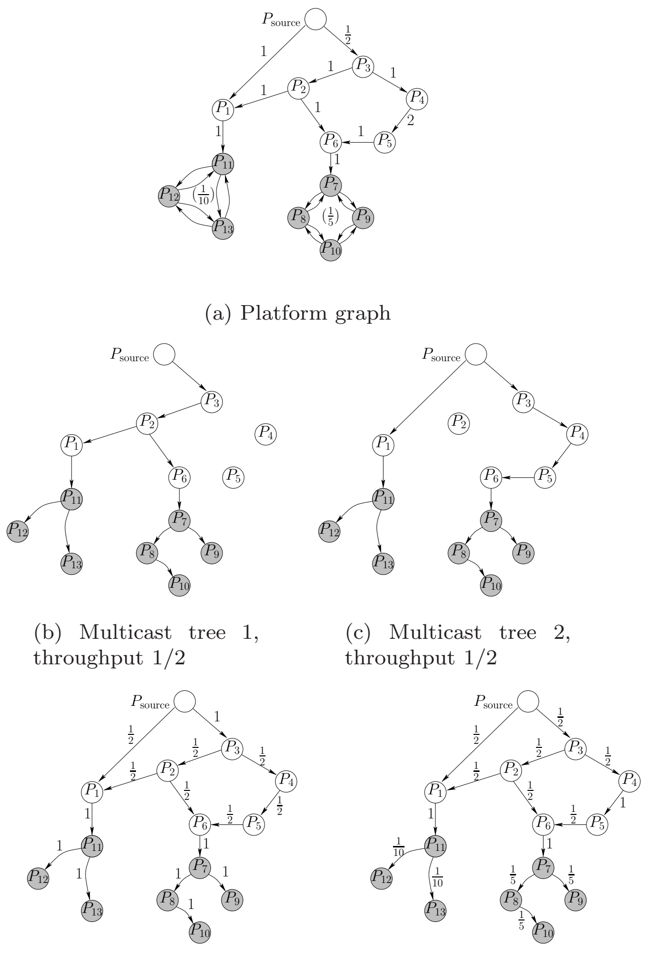
In this section, we work out a simple example. The platform graph is represented on Figure 1(a). The processor P_{source} aims at multicasting a series of messages to the target processors $P_7, P_8, \dots, P_{13}$ (which are shaded on the figure). Edges are labeled with the communication time needed to transfer one unit-size message. All edges between processors P_7, P_8, P_9, and P_{10} have weight 1/5, and edges between processors P_{11}, P_{12}, and P_{13} have weight 1/10.

Because the edge from P_6 to P_7 has weight 1, P_7 cannot receive more than one message per time-unit. This is an upper bound for the throughput that can be achieved with this platform for the Series of multicasts problem. In the following, we prove that this bound can be obtained, but only when using several multicast trees simultaneously.

Assume (by contradiction) that a single multicast tree $\mathcal{T}$ could deliver one message every time-unit. As P_{11} belongs to the set of target processors, P_1 has to receive the messages and to transfer them to P_{11}, so at least one of the edges (P_{source}, P_1) and (P_2, P_1) belongs to $\mathcal{T}$. Since $\mathcal{T}$ is a tree, only one of these edges belongs to $\mathcal{T}$. We examine both cases:

- $(P_{\text{source}}, P_1) \in \mathcal{T}$: then P_{source} sends a message to P_1 every time-unit, so it cannot perform any other sending operation. P_3 receives no message, and neither does P_7, hence a contradiction.

- $(P_2, P_1) \in \mathcal{T}$: then P_2 spends all its time sending messages to P_1. Therefore, P_2 has to receive its messages from P_3 at the same rate and P_6 has to receive its messages from P_5. As P_3 has to spend all its time sending data to P_2, P_4 (hence P_5 and P_6) cannot receive any message. Hence a contradiction.



(a) Platform graph

(b) Multicast tree 1, throughput 1/2

(c) Multicast tree 2, throughput 1/2

(d) Number of messages transferred along each edge within one time-unit

(e) Occupation time of each edge within one time-unit

Figure 1. Example for the Series **problem.**

Hence a throughput of one message every time-unit is not achievable with a single multicast tree. However, we outline an optimal schedule which reaches such a throughput, using two multicast trees. These trees, whose throughputs are both 1/2, are shown on Figures 1(b) and 1(c). The number of messages sent along each edge during on time-unit with this optimal solution is presented on Figure 1(d). Figure 1(e) shows the corresponding communication times on each edge. We point out that the two multicast trees are not edge-disjoint, but all the communications induced by each of them can be orchestrated so as to take place within one time-unit, as outlined in Figure 1(e). We see that some processors reach their maximum sending capacity, such as $P_{\text{source}}, P_1, P_2, P_3, P_4, P_6$; also, some processors reach their maximum receiving capacity: P_1, P_6, P_7, P_{11}.

4. Complexity results

In this section, we derive complexity results for the Series of Multicasts problem. We first show that even the simple problem to determine the optimal throughput that can be achieved for a succession of multicast operations 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 the paper [1].

4.1 Complexity of the Series of Multicasts problem

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 $G = (V, E, c)$, a source processor P_{source}, a set of destination processors $\mathcal{P}_{target}$, 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 $K \leqslant \log S$ and $\frac{K}{T} \geqslant \rho$?*

Theorem 1. *COMPACT-MULTICAST(G, P_{source}, $\mathcal{P}_{target}$, 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 $K \leqslant \log S$ 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 [4].

The proof of this result (available in [1]) 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 $G = (V, E, c)$, a source processor P_{source}, a set of destination processors $\mathcal{P}_{target}$, a rational bound for the throughput ρ, is there a periodic schedule consisting of $k \leqslant 2|E|$ multicast trees $\{T_1, \dots, T_k\}$, 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 $\forall i = 1, \dots, k,\ \log a_i + \log b_i \leqslant 4|E|(\log|E| + A = \log \max c_{i,j})$, and $\sum \alpha_i \geqslant \rho$?*

Theorem 3. *COMPACT-WEIGHTED-MULTICAST(G, P_{source}, $\mathcal{P}_{target}$, 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 $G = (V, E, c)$, a source processor P_{source}, a set of destination processors $\mathcal{P}_{target}$, if there exists a periodic schedule that achieves a throughput ρ, then there also exists a solution of COMPACT-WEIGHTED-MULTICAST(G, P_{source}, $\mathcal{P}_{target}$, ρ).*

The main two complexity results stated in this section should be compared to their equivalent for the broadcast problems.

	Broadcast	Multicast
The best tree	NP-hard [2]	NP-hard (Th. 1)
Combination of weighted trees	P [2]	NP-hard (Th. 3 and 4)

In many situation (e.g. the broadcast problem), using a relaxation such as the steady-state mode renders the problem much more simple. This relaxation is not sufficient for the multicast problem since the resulting optimization problem is NP-hard and does not even belong to the class APX. In [1], we show that these complexity results can be extended to a similar problem, namely Parallel Prefix computations.

5. LP-based heuristics

5.1. Lower and upper bound for multicast completion time

We consider a unit size message that can be arbitrarily split in smaller parts to be multicast on the platform. We denote by $x_i^{j,k},\ \forall P_i \in \mathcal{P}_{\text{target}},\ \forall (P_j, P_k) \in E$ the fraction of the message (of total size 1) from P_{source} to P_i that transits on the edge between P_j and P_k. For any node P_j, we denote by $\mathcal{N}^{\text{out}}(P_j)$ (resp. $\mathcal{N}^{\text{in}}(P_j)$) the set of nodes P_k such that $(P_j, P_k) \in E$ (resp. $(P_k, P_j) \in E$).

5.1.1 Set of general constraints

In what follows, we give a set of linear constraints that must be fulfilled by any solution.

- The first set of constraints states that the entire message has been sent from P_{source} and has been received at P_i:

$$(1) \quad \forall i \in \mathcal{P}_{\text{target}}, \qquad \sum_{P_j \in \mathcal{N}^{\text{out}}(P_{\text{source}})} x_i^{\text{source},j} = 1$$

$$(2) \quad \forall i \in \mathcal{P}_{\text{target}}, \qquad \sum_{P_j \in \mathcal{N}^{\text{in}}(P_i)} x_i^{j,i} = 1$$

- The second set of constraints states a conservation law at P_j, where $P_j \neq P_{\text{source}}$ and $P_j \neq P_i$ for the messages sent to P_i:

$$(3) \quad \begin{array}{c} \forall j, \quad P_j \neq P_{\text{source}} \text{ and } P_j \neq P_i, \\ \displaystyle\sum_{P_k \in \mathcal{N}^{\text{out}}(P_j)} x_i^{j,k} = \sum_{P_k \in \mathcal{N}^{\text{in}}(P_j)} x_i^{k,j} \end{array}$$

- The following set of constraints is related to the architectural framework of the platform. Let $n_{j,k}$ be the total fraction of packets that transit on the communication link between P_j and P_k. Let us suppose (until next section) that we know how to compute $n_{j,k}$. Therefore, the occupation time $t_{j,k}$ of the link (P_j, P_k) is given by

$$(4) \quad \forall (P_j, P_k) \in E, \qquad t_{j,k} = n_{j,k} \times c_{j,k}$$

We also need to write down the constraints stating that communication ports for both incoming and outgoing communications are not violated. The occupation time of the ports for incoming (resp. outgoing) communications will be denoted by $t_j^{(\text{in})}$ (resp. $t_j^{(\text{out})}$):

$$(5)\quad \forall j, \qquad t_j^{(\text{in})} = \sum_{P_k \in \mathcal{N}^{\text{in}}(P_j)} t_{k,j}$$

$$(6)\quad \forall j, \qquad t_j^{(\text{out})} = \sum_{P_k \in \mathcal{N}^{\text{out}}(P_j)} t_{j,k}$$

- The last set of constraint is related to the total multicast time T^* for a unit size message. The constraints simply state that T^* is larger than the occupation time of any incoming or outgoing communication port:

$$(7)\quad \forall j, \qquad T^* \geqslant t_j^{(\text{in})}$$

$$(8)\quad \forall j, \qquad T^* \geqslant t_j^{(\text{out})}$$

5.1.2 Total fraction of packets that transit on a communication link

We have denoted by $n_{j,k}$ the total fraction of packets that transit on the communication link between P_j and P_k. We know that a fraction $x_i^{j,k}$ of the message sent to P_i transit on this link. The main difficulty is that the messages transiting on this link and sent to different P_i's may well be partly the same, since the same overall message is sent to all the nodes in $\mathcal{P}_{\text{target}}$. Therefore, the constraint

$$(9)\quad n_{j,k} \quad = \sum_{P_i \in \mathcal{P}_{\text{target}}} x_i^{j,k}$$

that would hold true for a scatter operation, may be too pessimistic, but provides an upper bound for the completion time of the multicast. On the other hand, if our aim is to find a lower bound for the completion time of the multicast, we can make the optimistic assumption, stating that all the messages transiting between P_j and P_k are all sub-messages of the largest one, i.e.

$$(9')\quad n_{j,k} \quad = \max_{i \in \mathcal{P}_{\text{target}}} x_i^{j,k}$$

Therefore, the following linear program provides a lower bound for the multicast time of an infinitely divisible message of unit size:

Multicast-LB$(\mathcal{P}, \mathcal{P}_{\text{target}})$: Minimize T^*,
subject to Equations (1, 2, 3, 4, 5, 6, 7, 8, 9')

and the following linear program provides an upper bound for the multicast time of an infinitely divisible message of unit size:

Multicast-UB$(\mathcal{P}, \mathcal{P}_{\text{target}})$: Minimize T^*,
subject to Equations (1, 2, 3, 4, 5, 6, 7, 8, 9)

In the extended version of the paper [1], we show that neither the upper bound nor the lower one are tight, but we use the solution of the **Multicast-LB**$(\mathcal{P}, \mathcal{P}_{\text{target}})$ linear program in order to find an heuristic that differs at most by a factor $|\mathcal{P}_{\text{target}}|$ from the optimal solution, where $|\mathcal{P}_{\text{target}}|$ is the number of targets in the platform.

5.1.3 Broadcast on the whole platform

Another simple heuristic consists in performing a broadcast on the whole platform. Broadcast is a special case of multicast where the set of target nodes is the whole platform. Surprisingly, it has been proved that the optimal throughput given by the linear program **Multicast-LB**$(\mathcal{P}, \mathcal{P})$ (denoted by **Broadcast**$(\mathcal{P})$) can be achieved in this case. The construction of a schedule achieving this throughput relies on non-trivial graph theorems (weighted versions of Edmond's and König's theorems), and will not be detailed here. The interested reader may refer to [2] to find the description of such a schedule. The following set of inequalities holds true:

$$\textbf{Multicast-LB}(\mathcal{P}, \mathcal{P}_{\text{target}}) \leqslant \textbf{Multicast-UB}(\mathcal{P}, \mathcal{P}_{\text{target}})$$

$$\textbf{Multicast-LB}(\mathcal{P}, \mathcal{P}_{\text{target}}) \leqslant \textbf{Broadcast}(\mathcal{P})$$

$$\textbf{Multicast-LB}(\mathcal{P}, \mathcal{P}_{\text{target}}) \geqslant \frac{\textbf{Multicast-UB}(\mathcal{P}, \mathcal{P}_{\text{target}})}{|\mathcal{P}_{\text{target}}|}$$

5.2. Refined Heuristics

In this section, we briefly present three different heuristics based on the solutions given by **Broadcast**$(\mathcal{P})$, **Multicast-LB**$(\mathcal{P}, \mathcal{P}_{\text{target}})$ and **Multicast-UB**$(\mathcal{P}, \mathcal{P}_{\text{target}})$. The interested reader can find the formal description of all heuristics in [1]. Since we know how to build schedule from the solutions of **Broadcast**$(\mathcal{P})$ and **Multicast-UB**$(\mathcal{P}, \mathcal{P}_{\text{target}})$, the heuristics that we propose are all based on those solutions, on restricted or extended platforms.

5.2.1 Reduce-Broadcast

We start from **Broadcast**$(\mathcal{P})$ and try to reduce the broadcast platform. At each step of the algorithm, we select the node whose contribution to the propagation of the message in the lowest, that is the node with the minimum $\sum_{i\in\mathcal{P}_{\text{target}}}\sum_{P_j\in\mathcal{N}^{\text{in}}(P_m)} x_i^{j,m}$ in the solution of **Broadcast**($\mathcal{P}$). This node is discarded and we compute the broadcast time on the remaining platform graph $\mathcal{P}\setminus P_m$. Note that in this new platform, there may well be no path from P_{source} to a node in $\mathcal{P}_{\text{target}}$. In this case, we set **Broadcast**($\mathcal{P}\setminus P_m$)$= +\infty$. We stop pruning the platform graph when no improvement in the broadcast time can be found.

5.2.2 Augmented-Multicast

We start from **Multicast-LB**($\mathcal{P}, \mathcal{P}_{\text{target}}$) and aim at extending the set of target nodes $\mathcal{P}_{\text{target}}$ until broadcast is possible on the platform consisting of the nodes in $\mathcal{P}_{\text{target}}$. At each step of the algorithm, we select the node (not yet in $\mathcal{P}_{\text{target}}$) whose contribution to the propagation of the message is the largest. We stop adding nodes to $\mathcal{P}_{\text{target}}$ as soon as no improvement can be found.

5.2.3 Multisource-Multicast

The idea is to start from **Multicast-UB**($\mathcal{P}, \mathcal{P}_{\text{target}}$) and to augment the number of sources without changing the target nodes $\mathcal{P}_{\text{target}}$. The linear program **MulticastMultiSource-UB**($\mathcal{P}, \mathcal{P}_{\text{target}}, \mathcal{P}_{\text{source}}$) describes a multicast on the platform $\mathcal{P}$, with the set of target nodes $\mathcal{P}_{\text{target}}$ and the set of (ordered) intermediate sources $\mathcal{P}_{\text{source}} = \{P_{s_0}(= P_{\text{source}}), P_{s_1}, \dots, P_{s_l}\}$. In the linear program, $x_{s_i,j}^{k,l}$ denotes the fraction of the message sent to P_j, that was initially sent by source P_{s_i} and transiting on the communication link between P_k and P_l. We measure the occupation of communication links by summing up the different messages transiting on this link (Equation (9), corresponding to a scatter operation). Thus, it is possible to reconstruct a schedule from the solution of the linear program that achieves exactly the throughput given by the linear program.

$$
\begin{array}{l}
\textbf{MulticastMultiSource-UB}(\mathcal{P}, \mathcal{P}_{\text{target}}, \mathcal{P}_{\text{source}}) :\\
\textsc{Minimize } T^*,\\
\textsc{subject to}\\
\left\{
\begin{array}{l}
1, 2, 3, 9, 4, 5, 6, 7, 8, \text{ and}\\
(1b)\ \ \forall i \in \mathcal{P}_{\text{target}} \setminus \mathcal{P}_{\text{source}},\\
\qquad \sum_{j<l}\sum_{P_k\in\mathcal{N}^{\text{out}}(P_{s_j})} x_{s_j,i}^{s_j,k} = 1\\
(2b)\ \ \forall i \in \mathcal{P}_{\text{target}} \setminus \mathcal{P}_{\text{source}},\\
\qquad \sum_{j<l}\sum_{P_k\in\mathcal{N}^{\text{in}}(P_i)} x_{s_j,i}^{k,i} = 1\\
(3b)\ \ \forall i, k, j \leqslant l\ P_k \neq P_{s_j} \text{ and } P_i,\\
\qquad \sum_{P_l\in\mathcal{N}^{\text{in}}(P_k)} x_{s_j,i}^{l,k} = \sum_{P_l\in\mathcal{N}^{\text{out}}(P_k)} x_{s_j,i}^{k,l}
\end{array}
\right. .
\end{array}
$$

We start with $\mathcal{P}_{\text{source}} = \{P_{\text{source}}\}$. At each step of the algorithm, we select the node P_m from $\mathcal{P}\setminus\mathcal{P}_{\text{source}}$ such that $\sum_{i\in\mathcal{P}_{\text{target}}}\sum_{P_j\in\mathcal{N}^{\text{in}}(P_m)} x_i^{j,m}$ is maximal in the solution of **MulticastMultiSource-UB**($\mathcal{P}$,$\mathcal{P}_{\text{target}}$,$\mathcal{P}_{\text{source}}$). Since the contribution of P_m to the propagation of the message to the nodes of $\mathcal{P}_{\text{target}}$ is large, we can expect that adding it to the set of sources will decrease the multicast time. We stop adding new sources when no improvement in the multicast time can be found.

6. Tree-based heuristic

When targeting the problem of finding a good tree to multicast a message in a network, the most common goal is to optimize the resources consumed by the tree. Usually, a cost is associated to each communication link, and we aim at minimizing the cost of the multicast tree, that is the sum of the cost of its edges. This problem, called the *Minimum Steiner Tree*, is known to be NP-complete [5]. Several heuristics have been designed to approximate the solution of this problem, but none for the Series of Multicasts problem. Indeed, in this case, the goal is to build up a spanning tree of minimal cost, containing all target nodes, where the cost of a tree is the maximum sum, over all nodes in the tree, of the cost of the outgoing edges of that node. Indeed, for each node, the sum of the weights of outgoing edges is the time needed to forward the message to all its children.

A classical heuristic to build a minimum Steiner tree is the Minimum Cost Path Heuristic (first introduced in [9] and adapted to directed networks in [8]). In this algorithm, a tree (consisting initially of the source node of the multicast) grows until it spans all the multicast target nodes: at each step, we determine which target is the closest to the current tree, and we add the minimum path from the current tree to this target into the new tree.

From this heuristic designed for the Minimum Steiner Tree problem, we can derive a heuristic to our problem, although the metric is not the same. Consider a platform graph $G = (V, E, c)$ and a set of target nodes $\mathcal{P}_{\text{target}} = \{P_{t_1}, P_{t_2}, \dots, P_{t_{|T|}}\}$. We denote the multicast tree by *Tree*. The sketch of the algorithm is given in Figure 2.

We first choose the target node which is the closest, in the sense of our metric, to the current tree. This node, and the path to reach it, will be added to the tree. The only difficult part of the algorithm concerns the update of the cost of the edges (lines 13,14). On the resulting graph, the cost of any edge on the path from the source to any already added target node is equal to

```
MINIMUM-TREE($\mathcal{P}, \mathcal{P}_{\text{target}}$)
 1: $c(i,j) \leftarrow c_{i,j}$;
 2: $Tree \leftarrow (\{P_{\text{source}}\}, \emptyset)$;
 3: while $\mathcal{P}_{\text{target}} \neq \emptyset$; do
 4:   NodeToAdd$\leftarrow \emptyset$; path$\leftarrow \emptyset$; cost$\leftarrow \infty$;
 5:   for each node $P_t \in \mathcal{P}_{\text{target}}$ do
 6:     Compute the path $P(P_t)$ from $Tree$ to $P_t$ such that $c(P_t) = \max_{(i,j)\in P(P_t)} c(i,j)$ is minimal
 7:     if $c(P_t) < cost$ then
 8:       NodeToAdd$\leftarrow P_t$
 9:       path$\leftarrow P(P_t)$; cost$\leftarrow c(P_t)$;
10:   Add $P(P_t)$ and $P_t$ to the tree;
11:   $\mathcal{P}_{\text{target}} \leftarrow \mathcal{P}_{\text{target}} \setminus P_t$
12:   for each edge $(i,j) \in P(P_t)$ and all $k$ such that $(i,k) \in E$ do
13:     $c(i,k) \leftarrow c(i,k) + c(i,j)$;
14:     $c(i,j) \leftarrow 0$;
```

Figure 2. Tree-Based Heuristic

zero: for the selection of the next target, choosing edges which have already been chosen in the multicast tree of the message will not incur any additional cost, since these edges already carry the message. To explain line 13, consider a graph where the path $P_{\text{source}} \rightarrow \cdots \rightarrow P_i \rightarrow P_j \rightarrow \cdots \rightarrow P_{t_1}$ already belongs to the multicast tree. Assume we want to add the new target P_{t_2} to the multicast tree, using path $P_{\text{source}} \rightarrow \cdots \rightarrow P_i \rightarrow P_k \rightarrow \cdots \rightarrow P_{t_2}$. Since $P_{\text{source}}, \dots, P_i$ already belong to the multicast tree, there is no additional cost for using the corresponding edges. However P_i already spends $c(i,j)$ units of time sending data to P_j, so that P_i needs $c(i,j)+c(i,k)$ units of time to send the message to both nodes P_j and P_k. Thus, the potential cost induced by the edge (i,k) must be updated as shown at line 13.

7. Experimental results

In this section, we compare the heuristics given in this paper using simulations on ”realistic” topologies generated by Tiers, a random topology generator [3]. We perform the experiments for several numbers of nodes and targets. We use two types of configurations, one “small” platform type with 30 nodes and a “big” platform type with 65 nodes. For each type, 10 different platforms are generated using the same set of parameters. These platforms are used to test our heuristics with several densities of targets: the targets are randomly selected among the nodes belonging to the local area networks in the platforms. The results are presented on Figure 3. On these graphs, the name of the heuristics have the following meaning:

scatter This corresponds to the upper bound for the multicast completion time, as if the messages sent to each node were different. Figures 3(a) and 3(c) present the ratio of the results of the heuristics over this value.

lower bound This corresponds to the lower bound for the multicast completion time, which is not always achievable. Figures 3(b) and 3(d) present the ratio of the results of the heuristics over this value.

broadcast This consists in broadcasting to the whole platform, as described in Section 5.1.3.

MCPH The tree-based heuristic, adapted from the Minimum Cost Path Heuristic, and described in Figure 2.

Augm. MC, Red. MC and **Multisource MC** are the LP based heuristics developed in Section 5

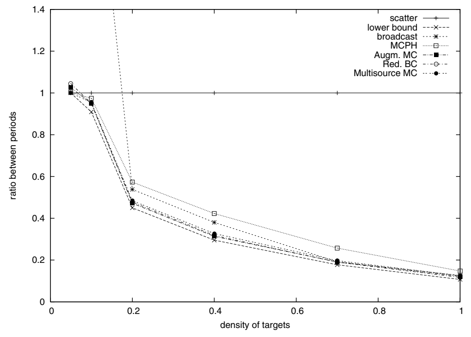
On Figure 3, we can see that the heuristics described in this paper are much closer to the lower bound than to the upper bound (scatter operation), except for the first experiment in a small platform, where the target nodes is reduced to one element. This is a very good result since the lower bound is not even guaranteed to be achievable.

The best results are given by the refined heuristics based on linear programming: **Augm. MC**, **Red. BC** and **Multisource MC**. However, the result of the tree-based heuristic **MCPH** is very close to their result, and its execution is shorter since it does not require to solve linear programs.

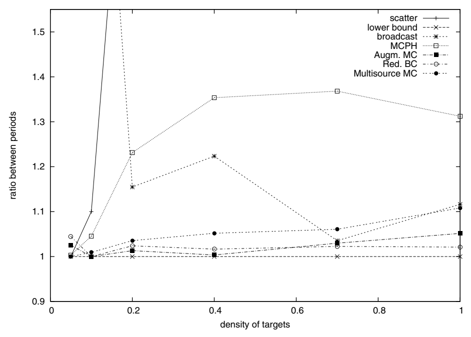
Surprisingly, we also notice that the result of the simple **broadcast** heuristic, included in our experiments for the sake of the comparison, performs well as soon as the density of targets among the local nodes is greater than 20%. To explain these good results, we recall that this heuristic does compute the optimal throughput of a broadcast on the whole platform. Moreover, the overall small size of the platform and the distribution of the target nodes leads to a platform graph such that there is almost one target node in each Local Area Network. That is a reason why the BROADCAST heuristic performs so well in this specific case.

8. Conclusion

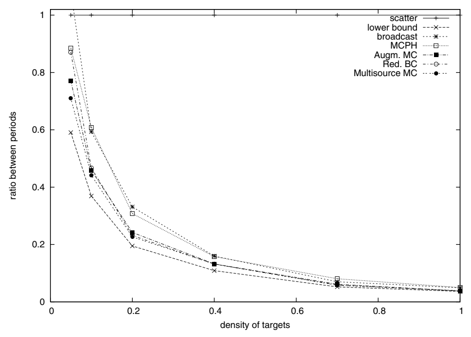
In this paper, we have studied the problem of multicasting a series of messages on heterogeneous platforms. Our major objective was to maximize the



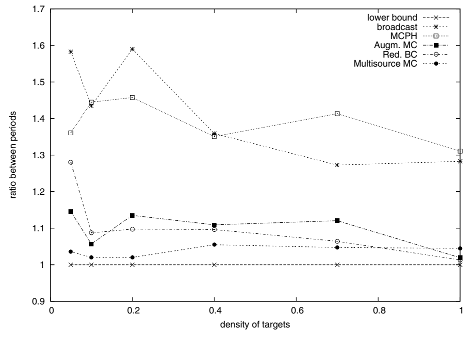
(a) Comparison with scatter on a small platform



(b) Comparison with the lower bound on a small platform



(c) Comparison with scatter on a big platform



(d) Comparison with the lower bound on a big platform

Figure 3. Comparison on the heuristics

throughput that can be achieved in steady-state mode, when a large number of same-size multicasts are executed in a pipeline fashion. Achieving the best throughput may well require that the target platform is used in totality: we have shown that using a single spanning tree is not powerful enough. But the general problem is very difficult: we have proved that determining the optimal throughput is a NP-complete problem. This negative result demonstrates that pipelining multicasts is more difficult than pipelining broadcasts, scatters or reduce operations, for which optimal polynomial algorithms have been introduced [2, 6].

We have introduced several heuristics to solve the pipelined multicast problem, most based on linear programming, and one adapted from a Steiner tree heuristic. These heuristics perform very well: there are close to the theoretical lower bound.

References

[1] O. Beaumont, A. Legrand, L. Marchal, and Y. Robert. Complexity results and heuristics for pipelined multicast operations on heterogeneous platforms. Technical report, LIP, ENS Lyon, France, Feb. 2004.

[2] 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. Extended version available as Research Report of LIP, ENS Lyon, France, at www.ens-lyon.fr/LIP.

[3] K. L. Calvert, M. B. Doar, and E. W. Zegura. Modeling internet topology. *IEEE Communications Magazine*, 35(6):160–163, June 1997.

[4] C. Hanen and A. Munier. Cyclic scheduling on parallel processors: an overview. In P. Chrétienne, E. G. Coffman, J. K. Lenstra, and Z. Liu, editors, *Scheduling Theory and its Applications*, pages 193–226. John Wiley & Sons, 1994.

[5] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103, NY, 1972. Plenum Press.

[6] A. Legrand, L. Marchal, and Y. Robert. Optimizing the steady-state throughput of scatter and reduce operations on heterogeneous platforms. In *APDCM'2004, 6th Workshop on Advances in Parallel and Distributed Computational Models*. IEEE Computer Society Press, 2004.

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

[8] S. Ramanathan. Multicast tree generation in networks with asymmetric links. *IEEE/ACM Transactions on Networking*, 4(4):558–568, 1996.

[9] H. Takashami and A. Matsuyama. An approximate solution for the steiner tree problem in graphs. *Intl. J. Math Educ. in Sci. and Technol.*, 14(1):15–23, 1983.