Laboratoire de l'Informatique du Parallélisme

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

Complexity results and heuristics for pipelined multicast operations on heterogeneous platforms

Olivier Beaumont,
Arnaud Legrand,
Loris Marchal,
Yves Robert

February 2004

Research Report N° 2004-07

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`

Complexity results and heuristics for pipelined multicast operations on heterogeneous platforms

Olivier Beaumont, Arnaud Legrand, Loris Marchal, Yves Robert

February 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 to several targets, known as the multicast operation. Rather than seeking to minimize the execution time of a single multicast, we focus on steady-state performance. We target heterogeneous platforms, modeled by a graph where resources 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; most of them are based on linear programming. We prove that some of these heuristics 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 the linear programming approach.

Keywords: Scheduling, steady-state, heterogeneous platforms, complexity

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 des primitives de communications collectives, comme par exemple le multicast (diffusion de données vers plusieurs cibles). Nous supposons qu'il y a un grand nombre de messages à diffuser et cherchons à optimiser le débit de telles opérations en régime permanent. La plate-forme hétérogène sous-jacente est modélisée par un graphe où les différentes ressources (processeurs, liens de communication) ont des vitesses différentes. Nous montrons que calculer le meilleur débit pour une opération de multicast est un problème NP-dur, alors le meilleur débit pour un broadcast (diffusion vers tous les nœuds de la plate-forme) peut être calculé en temps polynomial. Nous introduisons donc plusieurs heuristiques pour le problème du multicast, dont la plupart sont basées sur la programmation linéaire, et dont certaines sont garanties à un facteur près de l'optimal. Nous réalisons des simulations pour tester les performances de ces heuristiques et montrons que leur résultat est proche d'une borne supérieure théorique sur le débit que nous avons obtenue avec notre approche utilisant la programmation linéaire.

Mots-clés: Ordonnancement, régime permanent, plate-formes hétérogènes, complexité

1 Introduction

Multicasting is a key communication primitive in computer networks [20]. Lin and Ni [24] have published a survey paper where they consider different multicast algorithms operating under several network models; they explain the close relationships between multicast algorithms and Steiner trees (see [37] for an overview of Steiner problems). Several authors have discussed optimized multicast algorithms for a variety of parallel architectures, such as wormhole routed networks [31], cut-through routed networks [10], and networks of workstations [34]. Recently, the design of multicasting algorithms has been the focus of many papers, due to the advent of new technologies such as mobile [1], wireless [36], ad-hoc [14], and optical networks [38].

In this paper, we consider multicasting 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*. 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 [22, 21], and a series of broadcast operations [6, 5]. 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 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 modify these heuristics to cope with the new objective to be minimized: the sum of the weights of the outgoing edges of any vertex in the tree is a bound on the time needed by the vertex to forward a message in the one-port model, hence a bound on the throughput of the series of multicasts.

The rest of the paper is organized as follows. The next section (Section 2) is devoted to the formal specification of our series of multicasts problem, including the description of the target heterogeneous network, and of the operating mode of the resources. We work out a little example in Section 3, to emphasize the fact that using a single multicast tree does not lead to the optimal throughput in general. Then Section 4 is devoted to complexity results: we prove that determining the optimal throughput is a NP-hard problem. After this negative theoretical result, we proceed to the design of heuristics. We first deal with LP-based heuristics in Section 5. We are able to guarantee some of them as approximation algorithms. We investigate tree-based heuristics in Section 6. We report some experimental data in Section 7. We discuss related work in Section 8. Finally, we state some concluding remarks in Section 9.

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 $\mathcal{P}_{\text{target}} = \{P_{t_1}, \ldots, P_{t_N}\}$. 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, as discussed in Section 8. 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 $e_{j,k} : P_j \to P_k$ 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, 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 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 $\text{SERIES}(V, E, c, P_{\text{source}}, \mathcal{P}_{\text{target}})$, 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 $P_7, P_8, \ldots, P_{13}$ (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.

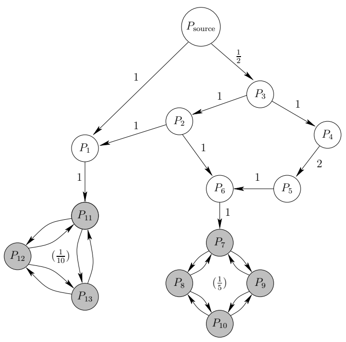
Because the edge from P_6 to P_7 has weight 1, P_7 cannot receive more than one message per time-unit. This is an upper bound for the throughput that can be achieved with this platform for the Series of multicasts problem. In the following, we prove that this bound can be obtained, but only when using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

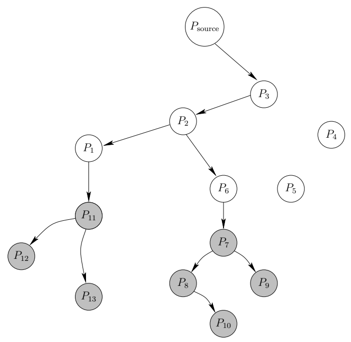
- $(P_{\text{source}}, P_1) \in \mathcal{T}$: then P_{source} sends a message to P_1 every time-unit, so it cannot perform any other sending operation. Hence P_3 receives no message, and neither does P_7, a contradiction.

- $(P_2, P_1) \in \mathcal{T}$: 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.

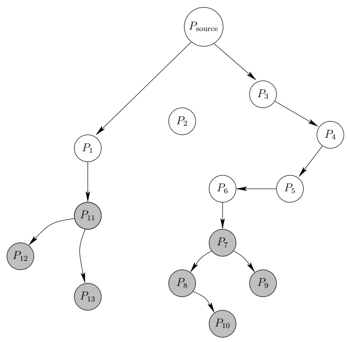
Hence a throughput of one message every time-unit is not achievable with a single multicast tree. However, we outline an optimal schedule which reaches such a throughput, using two multicast trees. These trees, whose throughputs are both 1/2, are shown on Figures 1(b) and 1(c). The number of messages 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}.



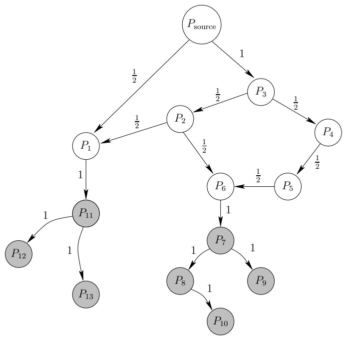
(a) Platform graph



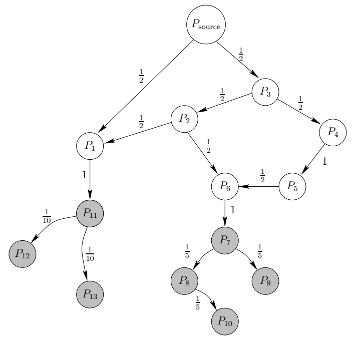
(b) Multicast tree 1, throughput 1/2



(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.

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). We conclude this section with a complexity result for the parallel prefix problem, using the same kind of a reduction as the one used for the multicast problem.

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($G, P_{source}, \mathcal{P}_{target}, \rho, S$)). *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($G, P_{source}, \mathcal{P}_{target}, \rho, S$) is NP-complete even for $S = 2$ (i.e. when using only one multicast tree).*

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 [16].

Proof. **COMPACT-MULTICAST $\in$ NP.** The first step is to prove that the COMPACT-MULTICAST problem does belong to NP. Given an instance $\mathcal{I}$ of the problem, we use a description of the K multicast-trees as the certificate. A multicast tree is simply defined as the list of the processors and edges involved in the propagation of a message emitted by P_{source} to all target processors. Clearly, if a given message is received twice by the same processor, we can suppress any one of the receptions, hence we can indeed restrict to trees for each message.

We claim that such a certificate can be checked in polynomial time: first, we check that each multicast tree is indeed rooted in P_{source}, has all processors in $\mathcal{P}_{target}$ as leaves, and is made up of valid edges from the platform graph. The most difficult point is to orchestrate the communications that take place in the trees.

For each processor P_i, we have the list of the trees that it belongs to, hence we can make a list of all the receptions and of all the emissions it has to execute; we derive recv_i, the amount of time P_i is receiving messages, as the sum of the time spent receiving along each edge (if the same edge appears in several multicast trees, we add up the numbers too: remember that we are using the one-port model). Similarly, we compute send_i, the amount of time P_i is receiving messages. We let the period T be the maximum of all theses quantities:

$$T = \max_{1 \leqslant i \leqslant p} \max(\text{recv}_i, \text{send}_i).$$

There is a nice theorem from graph theory that states that all the communications occurring in the K multicast trees can safely be scheduled within T time-units. This works as follows: from our platform graph G, we build a bipartite graph: for each node P_i in G, we create two nodes P_i^{send} and P_i^{recv}. For each communication from P_i to P_j in any multicast tree, we insert an edge between P_i^{send} and P_j^{recv}, which is weighted by the length of the communication. We are looking for a decomposition of this bipartite graph into a set of subgraphs where a node (sender or receiver) is occupied by at most one communication task. This means that at most one edge reaches each node in the subgraph. In other words, only communications corresponding to a matching in the bipartite graph can be performed simultaneously, and the desired decomposition of the graph is in fact an edge coloring. The weighted edge coloring algorithm of [32, vol.A chapter 20] provides in time $\mathcal{O}(|E|^2)$ a polynomial number of matchings, which are used to perform the different communications, and which provides the desired polynomial-size description of the schedule within a period.

Now, because there are no dependences between successive multicast operations, it is easy to finalize the full schedule, with a fixed number of periods (no more than the depth of the platform graph rooted at P_{source}) that corresponds to the initialization phase, and similarly for the clean-up phase.

The last step is to compute the ratio $\frac{K}{T}$ and to check that it does exceed the desired throughput ρ.

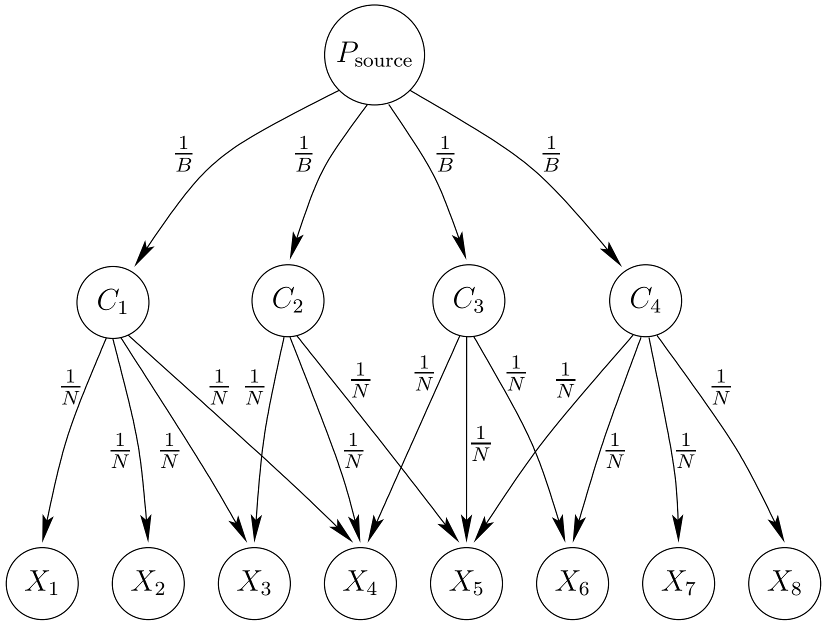


Figure 2: Instance $\mathcal{I}_2$ of COMPACT-MULTICAST obtained from the reduction of the following instance $\mathcal{I}_2$ of MINIMUM-SET-COVER: $X = \{X_1, \ldots, X_8\}$, $C = \{\{X_1, X_2, X_3, X_4\}, \{X_3, X_4, X_5\}, \{X_4, X_5, X_6\}, \{X_5, X_6, X_6, X_8\}\}$.

COMPACT-MULTICAST is complete. To prove the completeness of the COMPACT-MULTICAST problem, we use a reduction from the MINIMUM-SET-COVER problem, which is known to be NP-complete [12]. Consider an arbitrary instance $\mathcal{I}_1$ of MINIMUM-SET-COVER: let C be a collection of subsets of a finite set X of N elements, and let B, $1 \leqslant B \leqslant |C|$ be an integer bound. Does C contain a cover of X of size at most B? In other words, does there exist a subset C' of C of cardinal at most B such that each element of X belongs to at least one element of C'?

Given $\mathcal{I}_1$, we build the following instance $\mathcal{I}_2$ of COMPACT-MULTICAST. The platform graph $G = (V, E, c)$ is composed of (see Figure 2):

- a source processor P_{source},
- $|C|$ children C_i $(1 \leqslant i \leqslant |C|)$ of P_{source}, one for each element of C
- N leaf nodes X_i $(1 \leqslant i \leqslant N)$, one for each element in X. These leaf nodes are the N processors belonging to $\mathcal{P}_{\text{target}}$.

P_{source} is linked to each C_i by an edge of weight $1/B$. A node C_i is linked to X_j by an edge of weight $1/N$ if and only if $X_j \in C_i$. Finally, we let $S = 2$ and $\rho = 1$. Clearly, the size of $\mathcal{I}_2$ is polynomial in the size of $\mathcal{I}_1$. We have to prove that $\mathcal{I}_2$ has a solution if and only if $\mathcal{I}_1$ does:

- Assume first that $\mathcal{I}_1$ has a solution C' of cardinal at most B. Then we build a periodic schedule which uses a single multicast tree: $K = 1 = \log S$. During a period, P_{source} sends the same message to those children that belong to C'. P_{source} sequentially sends $|C'| \leqslant B$ messages on links whose weight is $1/B$: its total emission time is no more than one time-unit. We let the period be $T = 1$.

  Each processor C_j in C' receives one message per period. In steady-state, it forwards the message received during the previous period to its children. To avoid that a leaf receives the same message several times, we add the restriction that a leaf receives messages only from the leftmost node that belongs to C'. Formally, X_i receives messages from $C_j \in C'$ if $X_i \in C_j$ and there is no $k < j$ such that $X_i \in C_k$ and $C_k \in C'$.

  Because C' is a cover, each leaf receives a message per period, which requires $1/N \leqslant T$ time-units. Because each C_i has no more than N elements, its emission time is bounded by $N \times \frac{1}{N} = T$.

  We would use an actual period $T = \text{lcm}(N, B)$ to have integers, but this is just a scaling of the previous description. The throughput is one message per time-unit, hence $\rho = 1$, and $\mathcal{I}_2$ does have a solution.

- Assume now that $\mathcal{I}_2$ has a solution, i.e. a K-periodic schedule, where $K \leqslant \log S = 1$, of throughput at least $\rho = 1$. Then $K = 1$ and $T \leqslant 1$: there exists a single multicast tree in the schedule. Let C' be the set of C_j nodes that belong to the tree. The nodes in C' receive a message every T time-steps, and they forward these messages to all processors in $\mathcal{P}_{\text{target}}$: in other words, the subsets in C' form a cover of X. Because of the one-port constraint, the sum of the time spent by P_{source} to send messages to the nodes in C' is not greater then T, hence $|C'| \cdot \frac{1}{B} \leqslant T$. We derive that $|C'| \leqslant B$, hence $\mathcal{I}_1$ has a solution.

■

We can use the previous proof to derive an inapproximability result. The class APX is defined as the problems in NP which admits 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.*

Proof. Consider the (arbitrary) instance $\mathcal{I}_1$ of MINIMUM-SET-COVER. The instance $\mathcal{I}_2$ of COMPACT-MULTICAST built in the proof of Theorem 1 is restricted to using a single multicast tree. There is a direct one-to-one correspondence between covers in $\mathcal{I}_1$ and multicast trees in $\mathcal{I}_2$. More precisely, given any cover of cardinal K in $\mathcal{I}_1$, we can build a multicast tree of throughput $\frac{B}{K}$. Reciprocally, given a multicast tree of throughput ρ, using K nodes C_j, we derive that $\rho = \frac{B}{K}$ and a cover of cardinal K.

As a consequence, any polynomial-time λ-approximation algorithm for COMPACT-MULTICAST can be used to derive a polynomial-time λ-approximation algorithm for MININUM-SET-COVER. Therefore, all inapproximability results for the latter problem hold for the former. See [2] for such results. In particular, COMPACT-MULTICAST, just as MINIMUM-SET-COVER, 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($G, P_{source}, \mathcal{P}_{target}, \rho$)). *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 $\{T_1, \ldots, T_k\}$, where α_i is the average number of messages sent through tree T_i within one time-unit, $\alpha_i = a_i/b_i$, where a_i and b_i are integers such that $\forall i = 1, \ldots, k,\ \log a_i + \log b_i \leqslant 4|E|(\log|E| + \log \max c_{i,j})$, and $\sum \alpha_i \geqslant \rho$?*

Theorem 3. *COMPACT-WEIGHTED-MULTICAST($G, P_{source}, \mathcal{P}_{target}, \rho, S$) is NP-complete.*

The proof of this theorem is divided into two lemmas.

Lemma 1. *COMPACT-WEIGHTED-MULTICAST($G, P_{source}, \mathcal{P}_{target}, \rho$) $\in$ NP*

Proof. The proof of COMPACT-WEIGHTED-MULTICAST($G, P_{source}, \mathcal{P}_{target}, \rho$) $\in$ NP is similar to the proof of COMPACT-MULTICAST($G, P_{source}, \mathcal{P}_{target}, \rho, S$) $\in$ NP. We use a descrption of the k weighted multicast trees as a certificate.

We first check that each tree T_i is rooted in P_{source}, spans all processors in $\mathcal{P}_{target}$, and is made up of valid edges in the platform graph. Again, the difficult point is to check if the communications can be orchestrated. We first compute the least common multiple L of all b_i. Thus, $\alpha'_i = L \times a_i/b_i$ is the average (integer) number of messages sent through tree T_i during L time-units. As previously, we compute send_i (and recv_i), the amount of time spent by processor P_i to send data (and receive data) during L time-units. We let T be the maximum of these quantities:

$$T = \max_{1 \leqslant i \leqslant p} \max(\text{recv}_i, \text{send}_i).$$

We now use the same graph theorem as in proof of Theorem 1 to prove that all communications occuring in the multicast trees can be scheduled within T time-units. Thanks to the bound on $\log(a_i) + \log(b_i)$, we know that the α'_i's are polynomial in the size of the platform graph G, so the weighted edge coloring algorithm will produce a polynomial number of matchings, corresponding to a polynomial description of the schedule within the period. The last step is to check that the obtained throughput $\sum \alpha_i$ does exceed the bound ρ. ∎

Lemma 2. *COMPACT-WEIGHTED-MULTICAST($G, P_{source}, \mathcal{P}_{target}, \rho$) is complete.*

Proof. In order to show that MULTICAST is at least as difficult as any NP problem, we derive a polynomial reduction from MINIMUM-SET-COVER. Consider the same instance $\mathcal{I}_1$ as in the proof of Theorem 1. We use the same instance $\mathcal{I}_2$ of COMPACT-MULTICAST (whose size is polynomial in the size of $\mathcal{I}_1$), but without the bound S. We have to prove that $\mathcal{I}_2$ has a solution if and only if $\mathcal{I}_1$ does:

- Assume first that $\mathcal{I}_1$ has a solution C' of cardinal at most B. Then the periodic schedule built in the previous proof, with a single multicast tree, has the desired throughput $\rho = 1$, and $\mathcal{I}_2$ does has a solution.

- Assume now that $\mathcal{I}_2$ has a solution, i.e. there exists a set of k multicast trees $\{T_1, \ldots, T_k\}$ with average throughput $\alpha_1, \ldots, \alpha_k$ reaching the bound $\rho = 1$. Let L be the least common multiple of all b_i. We consider the schedule obtained while using the multicast trees in a period of L time-units. The (integer) number of messages sent through tree T_i is $\alpha'_i = L \times a_i/b_i$. In steady state, between time-steps t and $t+L$, at least L messages $M_1, \ldots, M_L$ are received by the leaf nodes $X_1, \ldots, X_N$, since the throughput is at least $\rho = 1$. Let c_i be the number of times that message M_i has been sent by P_{source}: because of the one-port constraint, $\sum_{i=1}^{L} c_i \cdot \frac{1}{B} \leqslant T$. Let $c_{i_0} = \min_{1 \leqslant i \leqslant L} c_i$. From the previous equation we have $Lc_{i_0}\frac{1}{B} \leqslant L$, hence $c_{i_0} \leqslant B$, which means that message M_{i_0} has been sent to at most B nodes in C. Let C' denote the set of these nodes. Because M_{i_0} must have been received by all leaf nodes, C' is a cover of X. We have the desired solution to $\mathcal{I}_1$. ∎

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 achieve a throughput ρ, then there also exists a solution of COMPACT-WEIGHTED-MULTICAST($G, P_{source}, \mathcal{P}_{target}, \rho$).*

Proof. In order to prove this result, we first derive a set of constraints that will be satisfied by any solution (periodic or not) to the multicast problem. Let us denote by $\mathcal{T}$ the set of multicast trees, i.e. the set of trees in G including all the nodes of $\mathcal{P}_{target}$. There may be an exponential number of such trees (with respect to the size of the graph), but the number of multicast trees is nevertheless finite. A solution of the MULTICAST problem is a set of weighted trees $\{(T_1, y_1), \ldots, (T_m, y_m)\}$, where $\mathcal{T} = \{T_1, \ldots, T_m\}$, which satisfy the following set of constraints:

$$\left\{\begin{array}{lll} (1,i) & \forall P_i, & \displaystyle\sum_{P_j \in \mathcal{N}^{\text{in}}(P_i)} \sum_{T_k \ni (P_j, P_i)} y_k c_{j,i} \leqslant 1 \\ (2,i) & \forall P_i, & \displaystyle\sum_{P_j \in \mathcal{N}^{\text{out}}(P_i)} \sum_{T_k \ni (P_i, P_j)} y_k c_{i,j} \leqslant 1 \\ (3,k) & \forall t_k \in \mathcal{T}, & y_k \geqslant 0 \end{array}\right.$$

Indeed, for any solution to the MULTICAST problem, 1-port constraints must be fulfilled, and from any solution of previous set of inequalities, one can derive a valid schedule (using the weighted version of König's theorem), where $\sum y_k$ message are multicast every time-unit. Thus, the solution of the following linear program provides an optimal solution of the

MULTICAST problem:

$$\begin{aligned}&\textbf{Maximize } \textstyle\sum_k y_k,\\ &\textbf{subject to}\\ &\begin{cases} (1,i) \quad \forall P_i, & \displaystyle\sum_{P_j \in \mathcal{N}^{\text{in}}(P_i)} \sum_{t_k \ni (P_j,P_i)} y_k c_{j,i} \leqslant 1 \\ (2,i) \quad \forall P_i, & \displaystyle\sum_{P_j \in \mathcal{N}^{\text{out}}(P_i)} \sum_{t_k \ni (P_i,P_j)} y_k c_{i,j} \leqslant 1 \\ (3,k) \quad \forall t_k \in \mathcal{T}, & y_k \geqslant 0 \end{cases}\end{aligned}$$

Let us denote by $\rho_{\max}$ the optimal value of the objective function. The previous linear program is of little practical interest since both the number of constraints and the number of variables are possibly exponential is the size of the original instance of the MULTICAST problem. Nevertheless, using linear programming theory [33], it is possible to prove that one of the optimal solution to the linear program is one instance of COMPACT-WEIGHTED-MULTICAST$(G, P_{source}, \mathcal{P}_{target}, \rho_{\max})$. Indeed, there is a vertex V of the polyhedron defined by linear constraints which is optimal, and V is given by the solution of a $|\mathcal{T}| \times |\mathcal{T}|$ linear system, such that at V, at least $|\mathcal{T}|$ inequalities among $|\mathcal{T}| + 2|E|$ are tight. Since only $2|E|$ constraints are not of the form $(3,k)$, we know that at least $|\mathcal{T}| - 2|E|$ constraints of the form $(3,k)$ are tight, i.e. that at most $2|E|$ trees have positive weight. Thus, there exist an optimal solution where at most $2|E|$ trees are actually used.
In order to achieve the proof of the theorem, we need to bound the size of the weights of those trees. Again, we consider the optimal solution defined by vertex V, which is given by the solution of a $|\mathcal{T}| \times |\mathcal{T}|$ linear system, where at most $m \leqslant 2|E|$ are not of the form $y_k = 0$. Let us consider the $m \times m$ linear system containing non-trivial equations. The coefficients of both the matrix and the right hand size are either 0, 1, or $c_{i,j}$. Let us set $y_i = \frac{a_i}{b_i}$. Both a_i and b_i can be computed using Cramer's rule, and therefore, both a_i and b_i are the determinant of matrices A_i and B_i whose coefficients belong to $\{0, 1, c_{i,j}\}$. Moreover,

$$\begin{array}{rcll} \det(A_i) & = & \prod_j \lambda_j & \text{where the } \lambda_j\text{'s are the eigenvalues of } A_i \\ & \leqslant & ||A_i||_2^m & \text{where } m \leqslant 2|E| \\ & \leqslant & (m \max c_{i,j})^m & \text{see [13]} \\ & \leqslant & (2|E| \max c_{i,j})^{2|E|} & \end{array}$$

Therefore

$$\log(a_i) \text{ and } \log(b_i) \leqslant 2|E|(\log(|E|) + \log(\max c_{i,j})),$$

and the sizes of both a_i and b_i satisfy the constraints of the instance COMPACT-WEIGHTED-MULTICAST$(G, P_{source}, \mathcal{P}_{target}, \rho_{\max})$.

Thus, among the optimal solution of the MULTICAST problem, there exist a solution which use at most $2|E|$ multicast trees, whose weights $\frac{a_i}{b_i}$ satisfy $\log a_i + \log b_i \leqslant 4|E|(\log(|E|) + \log(\max c_{i,j}))$. Therefore, if there exist a solution to the MULTICAST problem with throughput $\rho_{\max}$, then there exist a solution to COMPACT-WEIGHTED-MULTICAST$(G, P_{source}, \mathcal{P}_{target}, \rho_{\max})$. ■

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 [5]	NP-hard (Theorem 1)
Combination of weighted trees	P [6]	NP-hard (Theorems 3 and 4)

In many situation (e.g. the broadcast problem), using a relaxation such as the steady-state mode renders the problem much simpler. 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 the following section, we show that these complexity results can be extended to a similar problem: the Parallel Prefix computations.

4.2 Extension to the Parallel Prefix problem

Deriving the best throughput that can be achieved when pipelining macro-communications is not always as difficult as for the multicast problem. For scatter, broadcast, personalized all-to-all and reduce operations, it has been shown that the counterpart of the Series problem can indeed be solved in polynomial time. The general idea is to capture the steady-state operation in terms of a linear program, whose solution provides an upper bound of the achievable throughput. There remains to extract regular patterns from the linear program so as to derive a periodic schedule that achieves the bound. We come back to linear program formulations in Section 5, as the form the basis of some of our heuristics.

In this section, we show that the difficulty of the multicast problem is not an exception: we prove that optimizing the throughput of pipelined parallel prefix operations also is an NP-complete problem.

4.2.1 Definitions

We recall the sketch of a parallel prefix operation. Initially, some processors $P_0, \ldots, P_N$ own a local value $x_0, \ldots, x_N$. The goal is to compute the reduction of the first i values and to store it in P_i, for all indices $i \leqslant N$. At the end of the computation, P_i must hold $y_i = x_0 \oplus x_1 \oplus \cdots \oplus x_N$, where $\oplus$ is an associative, non-commutative[1] operator. This operation is widely used as a building block in the design of parallel algorithms: see [11] for a detailed list of applications.

The parallel prefix operation is more complex than a reduce operation, where only the last value y_N is to be computed and stored in some target processor. Intuitively, several partial reduce operations are to be interleaved and/or merged, and this dramatically increases the combinatorial nature of the problem.

We need to introduce some notations. Let $[k, m] = x_k \oplus \cdots \oplus x_m$ denote the partial reduction of x_k to x_m, for $0 \leqslant k, m \leqslant N$. Thus, $[i, i] = x_i$ and $[0, i] = y_i$ for $0 \leqslant i \leqslant N$. As $\oplus$ is associative, two partial results can be further reduced as follows:

$$[k, m] = [k, l] \oplus [l + 1, m] \text{ for } k \leqslant l < m$$

We let $T_{k,l,m}$ denote the computational task corresponding to this reduction, and $g(T_{k,l,m})$ its weight (proportional to the number of operations to be executed).

[1] When the operator is commutative, we have more freedom to assemble the final results. Of course it is always possible to perform the parallel prefix operation with a commutative operator, but without taking advantage of the commutativity.

As before, the architectural platform is represented by a directed graph $G = (V, E, c, w)$, enriched with processor computing powers: a processor $P \in V$ needs $g(T_{k,l,m}) \cdot w(P)$ time-units to execute $T_{k,l,m}$. Then, we need a function $f(k, m)$ to represent the size of the data element $[k, m]$, so that the time for P_i to send $[k, m]$ to P_j (along the edge $(P_i, P_j) \in E$ is $f(k, m) \cdot c_{i,j}$. We let $\mathcal{P} = \{P_0, \ldots, P_N\} \subset V$ denote the set of processors that participate in the parallel prefix operation. Finally, the whole platform/application graph is denoted as $(G, \mathcal{P}, f, g)$.

The counterpart of multicast trees will be prefix allocation schemes: starting from the initial values $[i, i]$, a scheme is the complete list of the computations and communications that take place to perform a whole parallel prefix. We use these schemes as certificates in the proof of NP-completeness.

4.2.2 Complexity

We formally state the decision problem associated to the determination of the optimal throughput when pipelining parallel prefix operations.

Definition 3 (COMPACT-PREFIX$(G, \mathcal{P}, f, g, \rho, S)$). *Given a weighted platform/application graph $(G, \mathcal{P}, f, g)$, where $G = (V, E, c)$, 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 parallel prefix operations every T units of time in steady-state, such that $K \leqslant \log S$ and $\frac{K}{T} \geqslant \rho$?*

Theorem 5. *COMPACT-PREFIX$(G, \mathcal{P}, f, g, \rho, S)$ is NP-complete.*

Proof. **COMPACT-PREFIX $\in$ NP.** The first step is to prove that the COMPACT-PREFIX problem does belong to NP. Given an instance $\mathcal{I}$ of the problem, we use a description of the K prefix allocation schemes as the certificate. We claim that such a certificate can be checked in polynomial time. As in the proof of Theorem 1, the most difficult part is to orchestrate the communications that take place in the different schemes. The proof is quite similar. There is an additional verification that the amount of time that each processor P spends computing tasks for the various schemes does not exceed the period T, but this is easy as it is equal to the sum of the time spent by P within each scheme.

COMPACT-PREFIX is complete. To prove the completeness of the COMPACT-PREFIX problem, we use a reduction from the MINIMUM-SET-COVER problem, as before. Consider an arbitrary instance $\mathcal{I}_1$ of MINIMUM-SET-COVER: let C be a collection of subsets of a finite set X of N elements, and let B, $1 \leqslant B \leqslant |C|$ be an integer bound. Does C contain a cover of X of size at most B? Let k be the cardinal of C. We let $C = \{C_1, \ldots, C_k\}$ and $X = \{X_1, \ldots, X_n\}$.

From $\mathcal{I}_1$ we construct the following instance $\mathcal{I}_2$ of COMPACT-PREFIX. The platform graph is composed of $2N + k + 1$ processors:

- a processor P_s
- k processors $C_1, \ldots, C_k$
- k edges from P_s to the C_i, each with a communication cost $1/B$
- N processors $X_1, \ldots, X_N$

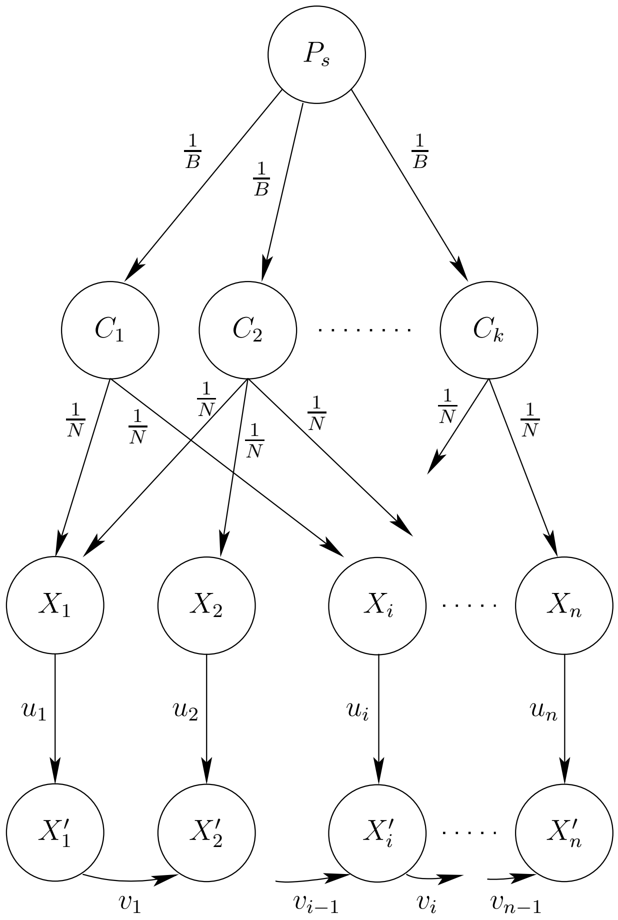


Figure 3: Platform graph for the instance $\mathcal{I}_2$

- there is an edge from C_i to X_j (of communication cost $1/N$) if and only if X_j belongs to C_i
- N processors $X'_1, \ldots, X'_N$
- for all i, there is an edge from X_i to X'_i, with communication cost $u_i = \dfrac{1}{i} - \dfrac{1}{(N+1)}$
- for all $i < N$, there is an edge from X'_i to X'_{i+1}; the communication cost of this edge is $v_i = \dfrac{1}{i+1} + \dfrac{1}{(N+1)\,i}$

This platform is represented on Figure 3. The set of processors taking part in the parallel prefix operation is $\mathcal{P} = \{P_s, X'_1, X'_2, \ldots, X'_N\}$ (more precisely, we should rename processors and denote this set as $\mathcal{P} = \{P_0, P_1, \ldots, P_N\}$, so that $P_0 = P_s$ and $P_i = X'_i$). We have unitary computation costs: we let $g \equiv 1$, i.e. $g(k,l,m) = 1$ for each task $T_{k,l,m}$. Each processor P belonging to $\mathcal{P}$ has the same computing power $w(P) = 1/N$, while the other processors do not compute at all ($w(P) = +\infty$ if $P \notin \mathcal{P}$). As for data sizes, we assume that the size of data elements are proportional to the length of the reduced interval: $f(i,j) = \text{size}([i,j]) = j-i+1$. Finally, we set $\rho = 1$ and $S = 2$. The size of $\mathcal{I}_2$ is clearly polynomial in the size of $\mathcal{I}_1$.

Next, we show that $\mathcal{I}_2$ admits a solution if and only if $\mathcal{I}_1$ does:

- Assume first that $\mathcal{I}_1$ has a solution: C' is a cover of size $|C'| \leqslant B$. We build a periodic schedule for the parallel prefix operation as follows. We use a single prefix allocation scheme (hence $K = 1 \leqslant \log S$):
  - During one time-unit period, P_s sends the message $[0,0]$ to those C_i that belong to C'. As $|C'| \leqslant B$, with links of capacity $1/B$, this does not last more than one time-unit.
  - During one period, each C_i such that $C_i \in C'$ sends the message $[0,0]$ to all X_j such that $X_j \in C_i$. As in the proof of Theorem 1, if X_j is covered by more than one C_i, then only the leftmost node sends the message. As C' is a cover, all the X_j get the message. Moreover, as each C_i sends the message to at most N nodes, using links of capacity $1/N$, the sending time does not exceed 1. Each X_j receives the message exactly once, the receiving time for these nodes is $1/N$.
  - During one period, X_j sends one message $[0,0]$ to X'_j in time $u_j \leqslant 1$
  - During one period, each node X'_i, where $i < N$, sends the i values $[1,1]$, $[2,2]$, $\ldots$, $[i,i]$ to node X'_{i+1}. The sending time for node X'_i is thus $i \times v_i = \frac{i}{i+1} + \frac{1}{(N+1)\,i} \leqslant 1$. The receiving time for node X'_1 is $u_1 = N/(N+1) \leqslant 1$. The receiving time for node X'_i, $i \geqslant 2$, is $u_i + (i-1) \times v_{i-1} = 1$
  - During one period, each node X'_i computes the reduction $y_i = (\cdots((x_0 \oplus x_1) \oplus x_2) \cdots) \oplus x_i$. The other nodes perform no computation. The computing time of node P_i is:
  $$\sum_{j=1}^{i} w(P_i) \times g(0,j,j) = \sum_{j=1}^{i} 1/n \times 1 = i/n \leqslant 1$$

  All these steps can be performed in a period of one time-unit, so we indeed reach throughput $\rho = 1$, hence a solution to $\mathcal{I}_2$.

- Assume now that $\mathcal{I}_2$ has a solution.
  - We first show that during a period T, $\rho \times T = T$ messages $x_0 = [0,0]$ (whose origin was in P_s) go through every edge $X_i \to X'_i$. Assume the contrary: there would exist an edge $X_i \to X'_i$ such that $R < T$ messages $[0,0]$ go through this edge. As X'_i performs T reductions $y_i = x_0 \oplus \cdots \oplus x_i$ per period, X'_i needs to receive at least T messages *"containing"* x_0: either messages of type $[0,0]$, or partially reduced messages $[0,j]$.

    There are two cases. Consider first the simple case where $i = 1$: if less than T messages go through edge $X_1 \to X'_1$, then X'_1 cannot receive enough messages $[0,0]$ to perform the reduction, because it has no other sources than X_1. Therefore, we assume $i \neq 1$ in the following.

    The $T - R$ missing messages for node X'_i can only be received from node from node X'_{i-1}. We point out that the size of a message $[j,k]$ produced by a partial reduction is the same as the size of the $(j-k+1)$ single value messages $[j,j]$, $[j+1,j+1]$, …, $[k,k]$. To simplify the notations (as computation is not taken into account for the moment), we consider, without loss of generality, that all messages involve single values: we replace every message $[j,k]$ in the solution by the corresponding single value messages $[j,j]$, $[j+1,j+1]$, …, $[k,k]$. Due to the shape of the platform graph, $[0,0]$ is the only message type which can reach X_i, hence which can be forwarded through the edge $X_i \to X'_i$. Let α_j be the quantity of messages $[j,j]$ going through edge $X'_{i-1} \to X'_i$ during one period. To perform T reductions $(x_0 \oplus \cdots \oplus x_i)$ in one period, node X'_i has to receive at least T messages of each type. It receives R messages $[0,0]$ from X_i, and no other messages on this edge. Overall, this leads to the following constraints:

    $$\begin{gathered} R + \alpha_0 \geqslant T \\ \forall 1 \leqslant j < i, \quad \alpha_j \geqslant T \end{gathered}$$

    The time spent by X'_i receiving data is the following:

    $$\begin{aligned} T^{\text{recv}}_{X'_i} &= R \times u_i + \left(\sum_{j=0}^{i-1} \alpha_i\right) \times v_{i-1} \\ &= N \times \left(\frac{1}{i} - \frac{1}{N+1}\right) + \left(\sum_{j=0}^{i-1} \alpha_i\right) \times \left(\frac{1}{i} + \frac{1}{(N+1)\,(i-1)}\right) \\ &\geqslant (T - \alpha_0) \times \left(\frac{1}{i} - \frac{1}{N+1}\right) + \big(T\,(i-1) + \alpha_0\big) \times \left(\frac{1}{i} + \frac{1}{(N+1)\,(i-1)}\right) \\ &= T + \alpha_0 \times \frac{i}{(N+1)\,(i-1)} \end{aligned}$$

    As $R < T$, $\alpha_0 \geqslant 1$ and X'_i has to receive data more than T time-units, a contradiction. So our assumption was false, and every node X_i does forward T messages $[0,0]$ to X'_i within one period.
  - We know that in a period (for example between time-steps t_0 and $t_0 + T$), T messages $[0,0]$ have been received by each X_i (and as many forwarded to X'_i). We

denote by $M_1, \ldots, M_T$ this set of messages. Let c_i be the number of nodes C_j that have received (and forwarded) the message M_i from node P_s. From the one-port constraint in emission for P_s, we know that:

$$\sum_{i=1}^{T} \frac{c_i}{B} \leqslant T$$

$$\text{so} \quad T \times \min_i c_i \times 1/B \leqslant T$$

This leads to $\min_i c_i \leqslant B$. We denote by i_{min} the index of the smallest c_i: $c_{i_{min}} \leqslant B$. The message $M_{i_{min}}$ has been transferred to all the nodes X_i through at most B nodes among the C_j. This gives a cover of X of cardinal less than or equal to B, so $\mathcal{I}_1$ has a solution. ∎

This is the counterpart of Theorem 1. Extensions to weitghed formulations can be derived similarly (all the counterparts of Theorem 2 to Theorem 4 hold for 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.

- Constraints at P_{source} and $P_i \in \mathcal{P}_{\text{target}}$
  The first set of constraints states that the entire message has been sent from P_{source} and has been received at P_i:

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

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

- Constraints at $P_j \neq P_{\text{source}}$ and $P_i \in \mathcal{P}_{\text{target}}$
  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:

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

- 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 in-coming and out-going communications are not violated. The occupation time of the ports for in-coming (resp. out-going) communications will be denoted by $t_j^{(\text{in})}$ (resp. $t_j^{(\text{out})}$):

$$\begin{aligned} (5) \quad \forall j, \qquad t_j^{(\text{in})} &= \sum_{P_k \in \mathcal{N}^{\text{in}}(P_j)} t_{k,j} \\ (6) \quad \forall j, \qquad t_j^{(\text{out})} &= \sum_{P_k \in \mathcal{N}^{\text{out}}(P_j)} t_{j,k} \end{aligned}$$

- The last set of constraint is related to the total multicast time T^* for a unit size message. The constraints simply state that the broadcast time T^* is larger than the occupation time of any edge and any in-coming or out-going communication port:

$$\begin{aligned} (7) \quad \forall j,k, \qquad & T^* \geqslant t_{j,k} \\ (8) \quad \forall j, \qquad & T^* \geqslant t_j^{(\text{in})} \\ (9) \quad \forall j, \qquad & T^* \geqslant t_j^{(\text{out})} \end{aligned}$$

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

$$(10) \quad n_{j,k} \quad = \sum_{P_i \in \mathcal{P}_{\text{target}}} x_i^{j,k}$$

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.

$$(10') \quad n_{j,k} \quad = \max_{i \in \mathcal{P}_{\text{target}}} x_i^{j,k}$$

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

$$\begin{array}{l}\textbf{Multicast-LB}(\mathcal{P},\mathcal{P}_{\text{target}}) : \text{MINIMIZE } T^*,\\ \text{SUBJECT TO Equations } (1, 2, 3, 4, 5, 6, 7, 8, 9, 10')\end{array}$$

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

$$\begin{array}{l}\textbf{Multicast-UB}(\mathcal{P},\mathcal{P}_{\text{target}}) : \text{MINIMIZE } T^*,\\ \text{SUBJECT TO Equations } (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)\end{array}$$

5.1.3 Distance between lower and upper bounds

Unfortunately, neither the upper bound nor the lower obtained above are tight, as shown in Figure 4. Nevertheless, we can 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. The approximation factor $|\mathcal{P}_{\text{target}}|$ is tight for the heuristic, as shown in Figure 5.

Proof. From the solution of the **Multicast-UB**$(\mathcal{P},\mathcal{P}_{\text{target}})$ linear program, a solution achieving the same throughput can be easily obtained. The interested reader may refer to [22, 21] where the reconstruction scheme from the linear program is presented. In this section, we prove that such a solution may differ at most by a factor $|\mathcal{P}_{\text{target}}|$ from the solution of the throughput given by the solution of **Multicast-LB**$(\mathcal{P},\mathcal{P}_{\text{target}})$, thus providing a heuristic with a guaranteed factor $|\mathcal{P}_{\text{target}}|$.

Let us the consider an optimal solution of the linear program **Multicast-LB**$(\mathcal{P},\mathcal{P}_{\text{target}})$, and let us denote by $x_i^{j,k}, n_{j,k}, t_{j,k}, T^*, t_j^{(\text{in})}$ and $t_j^{(\text{out})}$ the different quantities involved in the optimal solution. Then, clearly, the set of variables $x_i^{j,k}, n_{j,k}|\mathcal{P}_{\text{target}}|, t_{j,k}|\mathcal{P}_{\text{target}}|, T^*|\mathcal{P}_{\text{target}}|, t_j^{(\text{in})}|\mathcal{P}_{\text{target}}|$ and $t_j^{(\text{out})}|\mathcal{P}_{\text{target}}|$ is a solution to the linear program **Multicast-UB**$(\mathcal{P},\mathcal{P}_{\text{target}})$, whose completion time differs at most by a factor $|\mathcal{P}_{\text{target}}|$ from the optimal solution of **Multicast-LB**$(\mathcal{P},\mathcal{P}_{\text{target}})$. ■

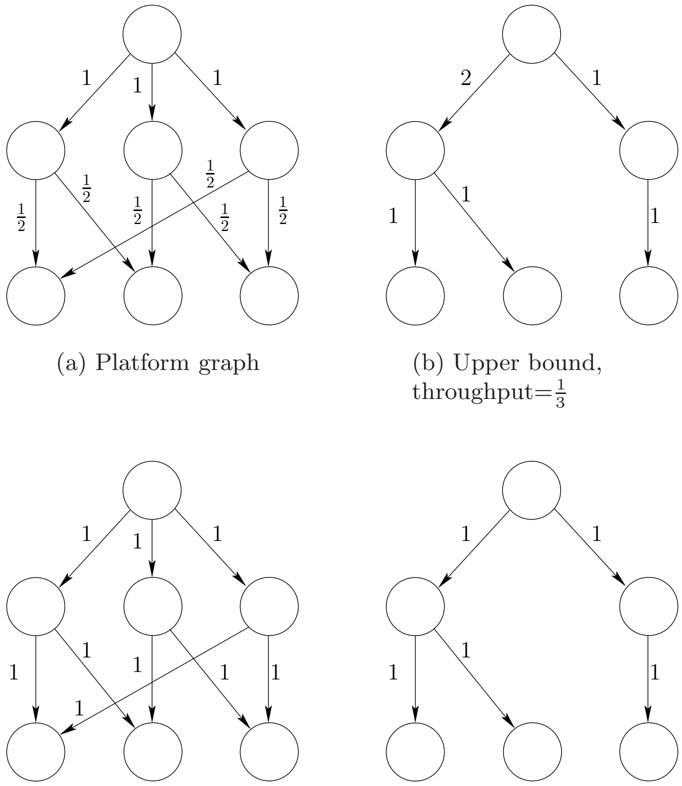
5.1.4 Broadcast of the whole platform

Another single 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**$(\mathcal{P},\mathcal{P})$=**Broadcast-EB**$(\mathcal{P})$ 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 [6, 5] to find the description of such a schedule. In what follows, we will denote by **Broadcast-EB**$(\mathcal{P})$ the optimal throughput of a broadcast on the whole platform. The following set of inequalities holds true:

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

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

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



(a) Platform graph

(b) Upper bound, throughput=$\frac{1}{3}$

(c) Lower bound, throughput=$\frac{2}{3}$

(d) Best multicast solution, throughput=$\frac{1}{2}$

Figure 4: None of the bounds are tight

5.2 Refined Heuristics

In this section, we present three different heuristics based on the solutions given by **Broadcast-EB**($\mathcal{P}$), **Multicast-LB**($\mathcal{P}, \mathcal{P}_{\text{target}}$) and **Multicast-UB**($\mathcal{P}, \mathcal{P}_{\text{target}}$). Since we know how to build schedule from the solutions of **Broadcast-EB**($\mathcal{P}$) 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 Starting from Broadcast-EB($\mathcal{P}$)

The idea is to reduce the broadcast platform. At each step of the algorithm, we select the node P_m from $\mathcal{P} \setminus \mathcal{P}_{\text{target}}$ such that

$$\sum_{i \in \mathcal{P}_{\text{target}}} \sum_{P_j \in \mathcal{N}^{\text{in}}(P_m)} x_i^{j,m}$$

is minimal in the solution of **Broadcast-EB**($\mathcal{P}$). Since the contribution of P_m to the propagation of the message to the nodes of $\mathcal{P}_{\text{target}}$ is low, we can expect that the broadcast time on $\mathcal{P} \setminus P_m$ will be lower. Note that in the platform $\mathcal{P} \setminus P_m$, there may be possibly no path from P_{source} to a node in $\mathcal{P}_{\text{target}}$. In this case, we set **Broadcast-EB**($\mathcal{P} \setminus P_m$)$= +\infty$. The sketch of the algorithm is given in Figure 6.

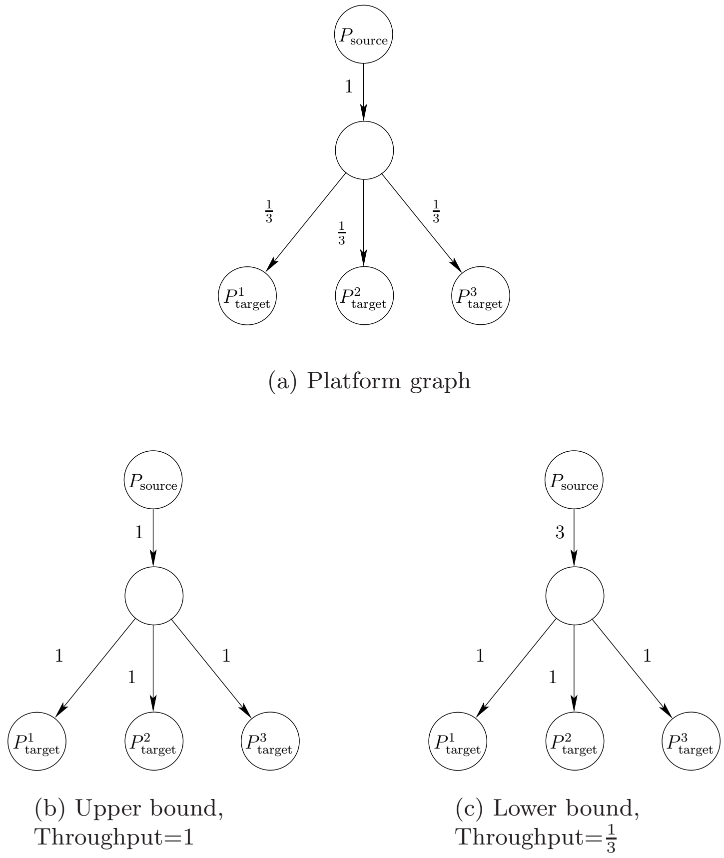


Figure 5: The distance between the lower and the upper bound may be $|\mathcal{P}_{\text{target}}|$

5.2.2 Starting from Multicast-LB($\mathcal{P}, \mathcal{P}_{\text{target}}$)

The idea is to extend the multicast platform $\mathcal{P}_{\text{target}}$ 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 P_m from $\mathcal{P} \setminus \mathcal{P}_{\text{target}}$ 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 **Multicast-LB**($\mathcal{P}, \mathcal{P}_{\text{target}}$). 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 the broadcast on $\mathcal{P}_{\text{target}} \cup P_m$ may be possible. Note that in the platform $\mathcal{P}_{\text{target}}$, there may be possibly no path from P_{source} to a node in $\mathcal{P}_{\text{target}}$. In this case, we set **Broadcast-EB**($\mathcal{P}_{\text{target}}$)$= +\infty$. The sketch of the algorithm is given in Figure 7.

5.2.3 Starting from Multicast-UB($\mathcal{P}, \mathcal{P}_{\text{target}}$)

The idea is to augment the number of sources without changing the multicast platform $\mathcal{P}_{\text{target}}$. We start with $\mathcal{P}_{\text{source}} = \{P_{\text{source}}\}$. At each step of the algorithm, we select the node P_m from

```
REDUCED_BROADCAST(𝒫)
Improvement ← YES
BestSol ← Broadcast-EB(𝒫)
while Improvement = YES do
  sort the nodes from 𝒫 \ 𝒫_target according to increasing values
    of Σ_{i∈𝒫_target} Σ_{P_j∈𝒩^in(P_m)} x_i^{j,m} in the solution of Broadcast-EB(𝒫):
    P_{k_1}, ..., P_{k_l}
  Improvement ← NO
  m ← 1
  while Improvement = NO and m ⩽ k_l do
    if Broadcast-EB(𝒫 \ P_m) ⩽ BestSol then
      Improvement ← YES
      BestSol ← Broadcast-EB(𝒫 \ P_m)
      𝒫 ← 𝒫 \ P_m
```

Figure 6: Heuristic based on Broadcast-EB($\mathcal{P}$)

```
AUGMENTED_MULTICAST(𝒫, 𝒫_target)
Improvement ← YES
BestSol ← Broadcast-EB(𝒫_target ∪ P_source)
while Improvement = YES do
  sort the nodes from 𝒫 \ 𝒫_target according to decreasing val-
    ues of Σ_{i∈𝒫_target} Σ_{P_j∈𝒩^in(P_m)} x_i^{j,m} in the solution of Multicast-
    LB(𝒫, 𝒫_target): P_{k_1}, ..., P_{k_l}
  Improvement ← NO
  m ← 1
  while Improvement = NO and m ⩽ k_l do
    if Broadcast-EB(𝒫_target ∪ P_source ∪ P_m) ⩽ BestSol then
      Improvement ← YES
      BestSol ← Broadcast-EB(𝒫_target ∪ P_source ∪ P_m)
      𝒫_target ← 𝒫_target ∪ P_m
```

Figure 7: Heuristic based on **Multicast-LB**($\mathcal{P}, \mathcal{P}_{\text{target}}$)

$\mathcal{P} \setminus \mathcal{P}_{\text{source}}$ 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 **Multicast-UB**$(\mathcal{P}, \mathcal{P}_{\text{target}})$. 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.

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 P_{s_i} and transiting on the communication link between P_k and P_l. Equation 1 states that intermediate source P_{s_i} will receive the entire message of overall size 1 from the intermediate sources $P_{s_0}, \dots, P_{s_{i-1}}$. Equation (1b) states that the message of overall size 1 sent to a node in $\mathcal{P}_{\text{target}} \setminus \mathcal{P}_{\text{source}}$ is the sum of the contributions of the different intermediate sources. We measure the occupation of communication links by summing up the different messages transiting on this link (Equation (10), corresponding to a scatter operation). Thus, it is easy to build up a schedule from the solution of the linear program that achieves exactly the throughput given by the linear program.

MulticastMultiSource-UB$(\mathcal{P}, \mathcal{P}_{\text{target}}, \mathcal{P}_{\text{source}})$:
Minimize T^*,
subject to

$$\left\{\begin{array}{ll}
(1)\ \ \forall s_i \in \mathcal{P}_{\text{source}}, & \sum_{j<i} \sum_{P_k \in \mathcal{N}^{\text{out}}(P_{s_j})} x_{s_j,s_i}^{s_j,k} = 1 \\
(1b)\ \ \forall i \in \mathcal{P}_{\text{target}} \setminus \mathcal{P}_{\text{source}}, & \sum_{j<l} \sum_{P_k \in \mathcal{N}^{\text{out}}(P_{s_j})} x_{s_j,i}^{s_j,k} = 1 \\
(2)\ \ \forall s_i \in \mathcal{P}_{\text{source}}, & \sum_{j<i} \sum_{P_k \in \mathcal{N}^{\text{in}}(P_{s_i})} x_{s_j,s_i}^{k,s_i} = 1 \\
(2b)\ \ \forall i \in \mathcal{P}_{\text{target}} \setminus \mathcal{P}_{\text{source}}, & \sum_{j<l} \sum_{P_k \in \mathcal{N}^{\text{in}}(P_i)} x_{s_j,i}^{k,i} = 1 \\
(3)\ \ \forall i,k,j \leqslant i\ P_k \neq P_{s_j} \text{ and } P_{s_i}, & \sum_{P_l \in \mathcal{N}^{\text{in}}(P_k)} x_{s_j,s_i}^{l,k} = \sum_{P_l \in \mathcal{N}^{\text{out}}(P_k)} x_{s_j,s_i}^{k,l} \\
(3b)\ \ \forall i,k,j \leqslant l\ P_k \neq P_{s_j} \text{ and } P_i, & \sum_{P_l \in \mathcal{N}^{\text{in}}(P_k)} x_{s_j,i}^{l,k} = \sum_{P_l \in \mathcal{N}^{\text{out}}(P_k)} x_{s_j,i}^{k,l} \\
(10)\ \ \forall (P_k,P_l) \in E, & n_{k,l} = \sum_{i \in \mathcal{P}_{\text{target}}} \sum_{j \in \mathcal{P}_{\text{source}}} x_{s_j,i}^{k,l} \\
(4)\ \ \forall (P_k,P_l) \in E, & t_{k,l} = n_{k,l} c_{k,l} \\
(5)\ \ \forall k, & t_k^{in} = \sum_{P_l \in \mathcal{N}^{\text{in}}(P_k)} t_{l,k} \\
(6)\ \ \forall k, & t_k^{out} = \sum_{P_l \in \mathcal{N}^{\text{out}}(P_k)} t_{k,l} \\
(7)\ \ \forall k,l, & T^* \geqslant t_{k,l} \\
(8)\ \ \forall k, & T^* \geqslant t_k^{in} \\
(9)\ \ \forall k, & T^* \geqslant t_k^{out}
\end{array}\right.$$

The sketch of the algorithm is given in Figure 8.

AUGMENTED_SOURCES($\mathcal{P}, \mathcal{P}_{\text{target}}, \mathcal{P}_{\text{source}}$)

Improvement $\leftarrow YES$
$\mathcal{P}_{\text{source}} \leftarrow \{P_{\text{source}}\}$
BestSol $\leftarrow$**MulticastMultiSource-LB**($\mathcal{P}, \mathcal{P}_{\text{target}}, \mathcal{P}_{\text{source}}$)
while Improvement$= YES$ **do**
   sort the nodes from $\mathcal{P} \setminus \mathcal{P}_{\text{source}}$ according to decreasing values of $\sum_{i \in \mathcal{P}_{\text{target}}} \sum_{P_j \in \mathcal{N}^{\text{in}}(P_m)} x_i^{j,m}$ in the solution of **MulticastMultiSource-LB**($\mathcal{P}, \mathcal{P}_{\text{target}}, \mathcal{P}_{\text{source}}$)): $P_{k_1}, \ldots, P_{k_l}$
   Improvement $\leftarrow NO$
   $m \leftarrow 1$
   while Improvement$= NO$ and $m \leqslant k_l$ **do**
      if **MulticastMultiSource-LB**($\mathcal{P}, \mathcal{P}_{\text{target}}, \mathcal{P}_{\text{source}} \cup P_m$) $\leqslant$ BestSol **then**
         Improvement $\leftarrow YES$
         BestSol $\leftarrow$ **MulticastMultiSource-LB**($\mathcal{P}, \mathcal{P}_{\text{target}}, \mathcal{P}_{\text{source}} \cup P_m$)
         $\mathcal{P}_{\text{source}} \leftarrow \mathcal{P}_{\text{source}} \cup P_m$

Figure 8: Heuristic based on **Multicast-UB**($\mathcal{P}, \mathcal{P}_{\text{target}}$)

6 Tree-based heuristics

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 [18]. 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 [35] and adapted to directed networks in [30]). 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. Let us consider a platform graph $G = (V, E, c)$ and a set of target nodes $\mathcal{P}_{\text{target}} = \{P_{t_1}, P_{t_2}, \ldots, P_{t_{|T|}}\}$. We denote the multicast tree by *Tree*. The sketch of the algorithm is given in Figure 9.

MINIMUM-TREE($\mathcal{P}, \mathcal{P}_{\text{target}}$)
1: $c(i,j) \leftarrow c_{i,j}$;
2: $\mathit{Tree} \leftarrow (\{P_{\text{source}}\}, \emptyset)$;
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** $c(P_t) < \mathit{cost}$ **then**
8: NodeToAdd$\leftarrow P_t$; path$\leftarrow P(P_t)$; cost$\leftarrow c(P_t)$;
9: Add $P(P_t)$ and P_t to the tree;
10: $\mathcal{P}_{\text{target}} \leftarrow \mathcal{P}_{\text{target}} \setminus P_t$
11: **for** each edge $(i,j) \in P(P_t)$ and all k such that $(i,k) \in E$ **do**
12: $c(i,k) \leftarrow c(i,k) + c(i,j)$;
13: $c(i,j) \leftarrow 0$;

Figure 9: Tree-Based Heuristic

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 12 to 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 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, let us consider the graph on Figure 10, where the path $P_{\text{source}} \to P_{t_1}$ already belongs to the multicast tree.

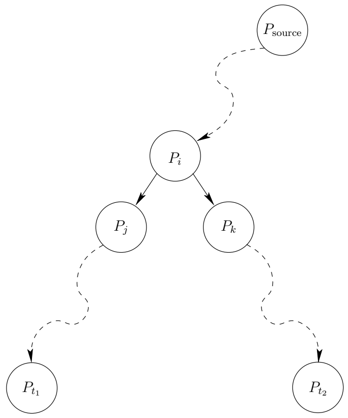


Figure 10: Example of the cost of the path $P_i \to P_{t_2}$ when $P_{\text{source}} \to P_i \to P_{t_1}$ is already in the tree.

Assume we want to add the new target P_{t_2} to the multicast tree, using path $P_{\text{source}} \to \cdots \to P_i \to \cdots \to P_{t_2}$. Since $P_{\text{source}}, \dots, P_i$ already belong to the multicast tree, there is no additional cost 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 generator of topology [9]. 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 (17 such nodes in the "small" platforms, and 47 nodes in the "big" platforms). The results are presented on Figure 11. 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 11(a) and 11(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 11(b) and 11(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 5.1.4.

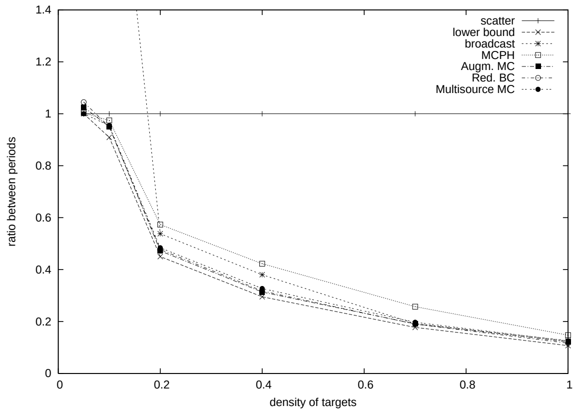
MCPH The tree-based heuristic, adapted from the Minimum Cost Path Heuristic, and described in Figure 9.

Augm. MC The AUGMENTED_MULTICAST heuristic (see Figure 7).

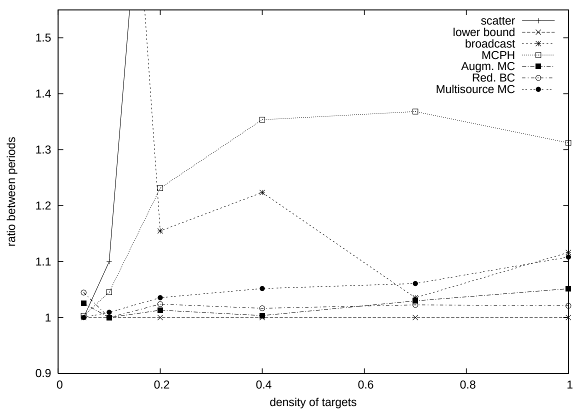
Red. BC The REDUCED_BROADCAST heuristic (see Figure 6).

Multisource MC The AUGMENTED_SOURCES heuristic, based on the linear program **MulticastMultiSource-UB** (see Figure 8).

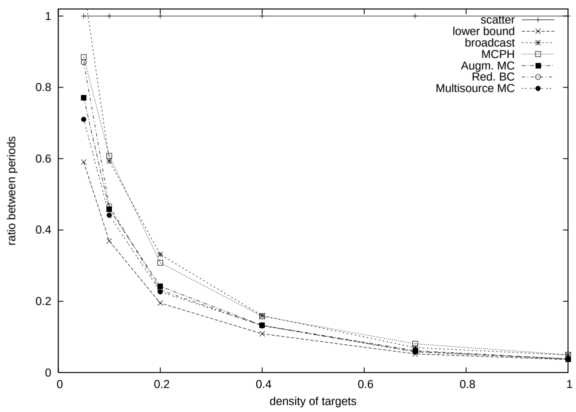
The last three heuristics are refined linear-programming heuristics described in Section 5.2.



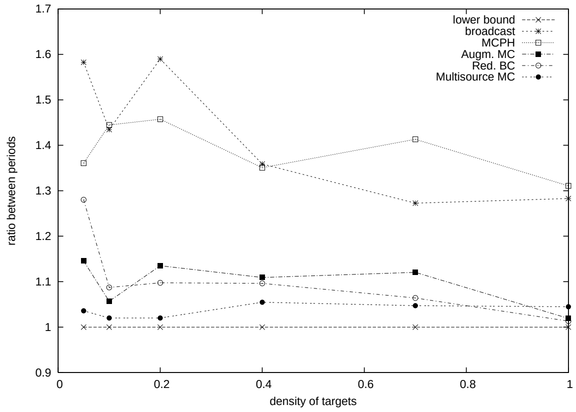
(a) Comparison with scatter on a small platform



(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 11: Comparison on the heuristics on two type of platforms

On Figure 11, we can see that the heuristics described in this paper are much closer to the lower bound than to the upper bound (scatter operation)[2]. This is 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

[2]except for the first experiment in a small platform, where the target nodes is reduced to one element

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 good in this specific case.

We work out an example on Figure 12. Figure 12(a) presents to topology of the platform, Figure 12(b) shows to spanning tree built by the **MCPH** heuristic, and Figure 12(c) the transfers involved in the solution of the **Multisource MC** heuristic. In this last figure, the secondary sources added by the algorithm are highlighted by a diamond shape, and we can notice that the resulting graph is not a tree: several paths may well be used to send different parts of the message.

8 Related work

Broadcasting and multicasting on heterogeneous platforms have been studied under different models, in the context of heterogeneous target platforms.

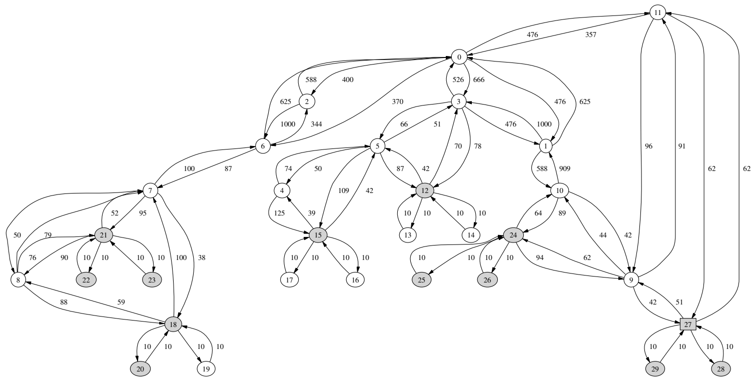
Banikazemi et al. [3] consider a simple model in which the heterogeneity among processors is characterized by the speed of the sending processors. In this model, the interconnection network is fully connected (a complete graph), and each processor P_i requires t_i time-units to send a (normalized) message to any other processor. The authors assert that this simple model of heterogeneity describes the different communication delays in a heterogeneous cluster. Some theoretical results (NP-completeness and approximation algorithms) have been proved for the problem of broadcasting a message under this model: see [15, 26, 25].

A more complex model is introduced in [4]: 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 cost, and a cost proportional to the length of the message.

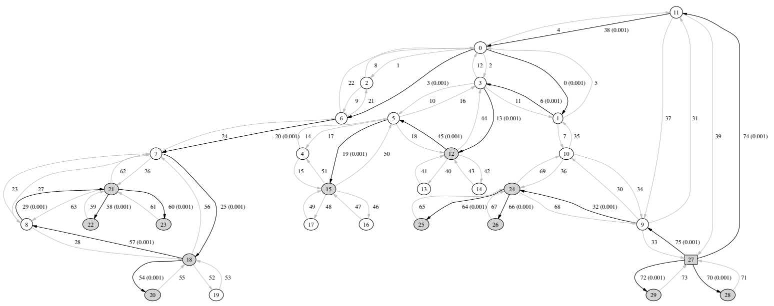
Yet another model of communication is introduced in [8, 7]: the time needed to transfer the message between any processor pair (P_i, P_j) is supposed to be divided into a start-up cost $T_{i,j}$ and a part depending on the size m of the message and the transmission rate $B_{i,j}$ between the two processors, $\frac{m}{B_{i,j}}$. 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 [8], some heuristics are proposed for the broadcast and the multicast using this model.

All previous models assume the *one port* protocol, which we used throughout this paper: a given processor can send data to at most one neighbor processor at a time. Usually, overlapping this operation with one reception (of independent data) is allowed.

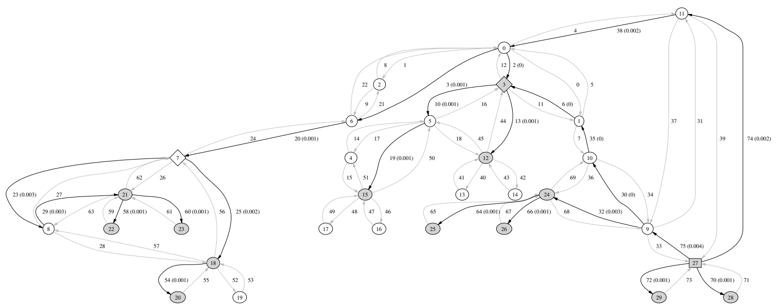
Generating multicast trees that optimize the overall cost of used resources has been well studied in order to minimize *tree cost* (or *Steiner cost*): we associate a cost to each communication link, and the tree cost is the sum of the cost of the links used in the tree. This problem, known as the *Minimum Steiner Tree*, is NP-complete [18]. Several polynomial-time heuristics exist, the most straightforward are the following. The Pruned Dijkstra Heuristic Heuristic consists in computing a shortest path tree for the full graph using Dijkstra algo-



(a) Topology graph generated by TIERS. The target nodes are filled in gray and the source node is the rectangular one. Edges are labeled with the time needed to transfer a unit-size message.



(b) Transfers involved in the **MCPH** solution. Edges are labeled with the quantity of messages transferred within one time-unit.



(c) Transfers involved in the **Multisource MC** solution. Edges are labeled with the quantity of messages transferred within one time-unit. Secondary sources (nodes 7 and 3) have a diamond shape.

Figure 12: Some graphs resulting of an experiment. With the **Multisource MC** heuristic, we need 789 time-units to broadcast a message to the targets, whereas 1000 time-units are used in the **MCPH** heuristic.

rithm and cutting the parts of the tree in which there is no target node. The Minimum Cost Path Heuristic [30, 35] grows a tree by selecting the "closest" target to the current tree; this heuristic inspired our MCPH heuristic in Section 6. The Distance Network Heuristic (or KMB heuristic) [19] starts by constructing a closure graph of the original network reducing the node set to the source and the targets. A minimum spanning tree is computed for this graph, and adapted to fit the original graph.

Other collective communications, such as scatter, all-to-all, gossiping and gather/reduce have been studied in the context of heterogeneous platforms: see [28, 17, 27, 23, 29] among others.

9 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 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 [6, 22]. In particular, although the broadcast and the multicast problems seem very similar (the target set is the only difference), there is a complexity gap between them: the best throughput to broadcast message to all nodes in the platform can be found in polynomial time, whereas finding the best throughput to multicast a message to a strict subset of these nodes is NP-hard.

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

[2] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation*. Springer, Berlin, Germany, 1999.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] Kenneth L. Calvert, Matthew B. Doar, and Ellen W. Zegura. Modeling internet topology. *IEEE Communications Magazine*, 35(6):160–163, June 1997. Available at http://citeseer.nj.nec.com/calvert97modeling.html.

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

[11] J. Reif (editor). *Synthesis of Parallel Algorithms*. Morgan Kaufmann, 1993.

[12] M. R. Garey and D. S. Johnson. *Computers and Intractability, a Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1991.

[13] G. H. Golub and C. F. Van Loan. *Matrix computations*. Johns Hopkins, 1989.

[14] T. Gopalsamy, M. Singhal, D. Panda, and P. Sadayappan. A reliable multicast algorithm for mobile ad hoc networks. In *ICDCS'02, 22nd Int. Conf. Distributed Computing Systems*, pages 563–570. IEEE Computer Society Press, 2002.

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

[16] 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.

[17] J.-I. Hatta and S. Shibusawa. Scheduling algorithms for efficient gather operations in distributed heterogeneous systems. In *2000 International Conference on Parallel Processing (ICPP'2000)*. IEEE Computer Society Press, 2000.

[18] 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.

[19] L.T. Kou, G. Markowsky, and L. Berman. A fast algorithm for steiner trees. *Acta Informatica*, 15:141–145, 1981.

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

[21] 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.

[22] 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.

[23] 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.

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

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

[26] 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.

[27] 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.

[28] B. Lowekamp and A. Beguelin. Eco: Efficient collective operations for communication on heterogeneous networks. In *10th International Parallel and Distributed Processing Symposium (IPDPS'96)*. IEEE Computer Society Press, 1996.

[29] F. Ooshita, S. Matsumae, and T. Masuzawa. Efficient gather operation in heterogeneous cluster systems. In *Proceedings of the 16th International Symposium on High Performance Computing Systems and Applications (HPCS'02)*. IEEE Computer Society Press, 2002.

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

[31] D.F. Robinson, P.K. McKinley, and B.H.C. Cheng. Optimal multicast communication in wormhole-routed torus networks. *IEEE Trans. Parallel Distributed Systems*, 6(10):1029–1042, 1995.

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

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

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

[35] 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.

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

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

[38] Y. Yang, J. Wang, and C. Qiao. Nonblocking WDM multicast switching networks. *IEEE Trans. Parallel Distributed Systems*, 11(12):1274–1287, 2000.