Theory Comput. Systems **35**, 641–665 (2002)
DOI: 10.1007/s00224-002-1047-5

Theory of Computing Systems

© 2002 Springer-Verlag
New York Inc.

Polynomial-Time Algorithms for Minimum-Time Broadcast in Trees*

Johanne Cohen,[1] Pierre Fraigniaud,[2] and Margarida Mitjana[3]

[1]LORIA, Université Nancy,
54003 Nancy cedex, France
Johanne.Cohen@loria.fr

[2]LRI-CNRS, bât. 490, Université Paris-Sud,
91405 Orsay cedex, France
Pierre.Fraigniaud@lri.fr

[3]Departamento Matemàtica Aplicada,
Universidad Politècnica de Catalunya,
Barcelona, Spain
margarida@ma1.upc.es

Abstract. This paper addresses the minimum-time broadcast problem under several modes of the *line* model, i.e., when long-distance calls can be placed along paths in the network. It is well known that the minimum-time broadcast problem can be solved in polynomial time under the single-port edge-disjoint paths mode. However, it is equally well known that either relaxing the model to the *all-port* edge-disjoint paths mode, or constraining the model to the single-port *vertex-disjoint* paths mode, leads to NP-complete problems; and exact solutions have been derived for specific topologies only (e.g., hypercubes or tori). In this paper we present polynomial-time algorithms for minimum-time broadcast in trees. These algorithms are obtained by application of an original technique called the *merging method*, which can be applied in a larger context, for instance, to solve the multicast problem or to address the restricted regimen. The merging method requires solving the *minimal contention-free matrix problem* whose solution presents some interest on its own.

* Preliminary versions of this paper have been presented at the 10th ACM Symposium on Parallel Algorithms and Architectures (SPAA '98) [9], at the 10th ACM–SIAM Symposium on Discrete Algorithms (SODA '99) [14], and at the DIMACS workshop on Robust Communication Networks [13].

1. Introduction

Recent advances in telecommunication systems enhanced standard point-to-point communication protocols to multipoint protocols. These latter protocols are of particular interest for group applications. Those groups involve more than two users (some may even involve thousands of users) sharing a common application, e.g., video-conferences, distributed data-bases, media-spaces, games, etc. Several protocols have been proposed to handle and to control a large group of users [16], [35]. A common feature of these protocols is to use trees as underlying network topology, either a single tree connecting all the group members (e.g., the protocol CBT [1]) or several trees, one for every source (e.g., the protocol PIM [15]). The traffic between the members is then routed along the edges of the tree(s), from the source (or core) to the members of the group. Technologies such as circuit-switching and wormhole [17], ATM virtual path [18], or single-hop WDM optical systems [3] provide distance-insensitive communications. A (physical or virtual) path is established between two participants of a communication session, and data flow through this path, cutting through intermediate nodes which act as transmitters, and do not have access to the content of these data. This avoids costly receive and send operations such as in store-and-forward technology, and the cost of the communication transfer is almost invariant with the distance between the two members of the session. Motivated by these facts, this paper addresses the *broadcast* problem in *trees* under the *line* communication model. We provide centralized and static solutions for this problem. This is clearly far from practical applications since multipoint protocols must be dynamic and, of course, decentralized. However, beside their algorithmic interests, our solutions allow the efficiency of distributed multipoint protocols to be measured. In other words, multipoint protocols are based on on-line algorithms, and analyzing these algorithms requires the knowledge of an optimal off-line solution. This paper provides such off-line solutions.

More formally, the broadcast problem is defined as follows. We are given a graph $G = (V, E)$, and a source node $s \in V$. The objective is to compute a communication protocol ensuring the dissemination of atomic information originally placed at s to all the other nodes of the graph. The structure of the protocol depends on graph topology and on the communication model. It is generally assumed that communications proceed by a sequence of simultaneous *calls* of duration 1. During a given call, the source of the call (the "caller") sends the information to the destination of the call (the "callee"). Two main families of communication models have been investigated: the *local* model and the *line* model. The local model specifies that the caller and the callee must be two neighboring nodes in the graph. In contrast, in the line model a call is a path connecting a caller to a callee. This paper is concerned with this latter model which, again, captures distance-insensitive technologies such as circuit-switching, wormhole, virtual path in ATM, and single-hop WDM routing in all-optical networks. (On the other hand, store-and-forward routing is obviously better modeled by the local model, which incorporates the internodes distances and hence captures the latency of the system induced by its diameter.)

Several variants of the line model can be distinguished. The *edge-disjoint* (resp. *vertex-disjoint*) paths mode specifies that simultaneous calls must be pairwise edge-disjoint (resp. vertex-disjoint). The *single-port* constraint specifies that a node can par-

ticipate in at most one call at a time, either as a caller or as a callee. At the other extreme, the *all-port* constraint specifies that a node can simultaneously participate (as a caller) in as many calls as its out-degree. (Note that, since the information is atomic, there is no need for a node to behave twice as a callee.)

The efficiency of a communication protocol is estimated in terms of the number of *rounds*. Round t, $t = 1, 2, \ldots$, is defined as the set of all calls performed between time $t-1$ and time t. A broadcast protocol from s to G performing in k rounds is optimal if there is no protocol performing in less than k rounds that achieves a broadcast from s to G. The number of rounds of an optimal broadcast protocol from s to G under the communication model $\mathcal{M}$ is denoted by $b_{\mathcal{M}}(G, s)$. The complexity of the broadcast problem depends heavily on $\mathcal{M}$. To summarize, once $\mathcal{M}$ is fixed, the broadcast problem with respect to the communication model $\mathcal{M}$ can be reformulated as follows: We are given a graph $G = (V, E)$ and a source node $s \in V$. The objective is to compute a broadcast protocol from s to G performing in $b_{\mathcal{M}}(G, s)$ rounds under the communication model $\mathcal{M}$.

The broadcast problem has been investigated in the literature since the early 50s (see in particular the survey [27]). With the growing interest in both parallel systems and telecommunication systems, a huge literature from the late 80s to the late 90s has been devoted to specific group-communication problems and specific network topologies (see the surveys [24] and [28]). Most of these contributions were derived for the single-port local model. Since the broadcast problem is NP-complete under this model [34], [40], several approximation algorithms [2], [19], [32], [38] and heuristics [25], [39] have been proposed. On the other hand, it was shown that computing an optimal broadcast protocol in a tree under the single-port local model is polynomial [37], and a linear-time algorithm returning the set of nodes with minimal broadcast-time has been derived [40]. The multiple-source broadcast problem in trees has also been studied [21], and, to complete this brief survey on trees, it was shown [26], [33] that, for any n, there exists a tree whose broadcast-time from any source is at most $\log_\varrho n$ rounds, where $\varrho = (1+\sqrt{5})/2$, and that it is the best that can be achieved. From all these results, one can say that the broadcast problem in trees under the single-port local model is solved.

The situation is different when long-distance calls are allowed, that is, under the variants of the line model. (Note that, in the line model, as opposed to [7] and [8], intermediate nodes along a path joining a caller to a callee do not receive copy of the transmitted information which cut through the routers of these nodes.)

In the single-port edge-disjoint paths mode, it was shown that every undirected n-node graph has a broadcast time $\lceil \log_2 n \rceil$ (see [20], and also [31]), and an optimal broadcast protocol can be computed in polynomial time. Interestingly, this result can be extended to the case in which the paths (along which the calls are performed) are set up by a shortest path routing function (see [12]). In contrast, under the same model, the broadcast problem is NP-complete for digraphs [10]. An $O(d/\log d)$-approximation algorithm for digraphs of maximum degree d has been derived in [23].

The broadcast problem is NP-complete for both graphs and digraphs under the single-port vertex-disjoint paths mode [10]. It was hence studied for specific topologies such as cycles and tori [29], [30]. An $O(\log n/\log\log n)$-approximation algorithm for graphs has been derived [32]. This approximation ratio has recently been slightly improved to $O(\log n/\log OPT)$ in [22]. For trees, results differ depending on whether

the tree is directed or not (a directed tree is a tree whose edges are directed toward the leaves, i.e., no "upward" calls are possible). For undirected trees, the best known result is a 3-approximation algorithm [22], whereas, for directed trees, there is an $O(n^3)$-time algorithm which returns an optimal broadcast protocol [5], [6].

Finally, the broadcast problem is also NP-complete under the all-port edge-disjoint paths mode [10]. Again, some results have been derived for specific topologies such as tori and hypercubes [4], [36]. Moreover, an $O(\log d)$-approximation algorithm for graphs and digraphs of maximum degree d has been derived in [23].

In this paper we describe a new technique to compute optimal broadcast protocols in trees under several variants of the line model. This technique consists merely of "merging" several protocols designed for subtrees. The protocols are described by their "shadow," i.e., a binary vector. The way the shadows are merged depends on the communication mode. We illustrate this technique by giving:

1. An $O(n \log n)$-time algorithm which, given any directed tree T rooted at s, returns an optimal broadcast protocol from s to T under the all-port edge-disjoint paths mode. This algorithm extends to undirected trees and to the multicast problem (i.e., the destination nodes form a proper subset of the nodes).
2. An $O(n^2)$-time algorithm which, given any directed tree T rooted at s, returns an optimal broadcast protocol from s to T under the single-port vertex-disjoint paths mode. This latter result improves the complexity of the previously known algorithm by a factor of $\Theta(n)$. This algorithm extends to the multicast problem

Both algorithms extend to the restricted regimen in which only specified nodes are allowed to relay messages. At the current time, the situation can hence be summarized as follows:

	Local model	Single-port edge-disjoint	All port edge-disjoint	Single-port vertex-disjoint
Trees	[37]	[12], [20]	Theorem 2	3-approximation [22]
Directed trees	[37]	2-approximation [Corollary 1]	Theorem 1	[5], [6], and Theorem 4

Finding exact solutions for the single-port edge-disjoint paths mode in directed trees, as well as for the single-port vertex-disjoint paths mode in undirected trees, seems to require some more work since, as is shown later, our technique would require some extension to fit with the corresponding two models. We were not able to produce such an extension, and we hence left these two cases as open problems. Note however that we derive a 2-approximation algorithm for the single-port edge-disjoint paths mode in directed trees (see Corollary 1), and a 3-approximation algorithm for single-port vertex-disjoint paths mode in (undirected) trees is presented in [22]. There also remains the question of whether linear-time algorithms can be designed for the broadcast problem. Our algorithms have time-complexity $O(n \log n)$ for the all-port model, and $O(n^2)$ for the single-port model.

The paper is organized as follows. The next section introduces some notions and preliminary results that will be useful for our main theorems. Then Section 3 addresses the all-port edge-disjoint paths mode, and shows how protocols can be merged in this mode. Section 4 addresses the so-called "Contention-Free Matrix Problem" whose resolution

will be shown to be helpful in solving the broadcast problem under the single-port vertex-disjoint paths mode. This is done in Section 5. Finally, Section 6 contains some concluding remarks on the extension of our results to the multicast problem, and to the restricted regimen.

2. Shadows and Lexical Optimality

A broadcast protocol $\mathcal{B}$ from a node s of a directed tree T rooted at s can be described by the list of calls performed by $\mathcal{B}$ at every round. Our constructions of optimal broadcast protocols are based on the following definition.

Definition 1. Let $\mathcal{B}$ be a broadcast protocol from s performing in r rounds in a directed tree $T = (V, E)$ rooted at s. The *shadow* of $\mathcal{B}$ on an edge $e \in E$ is the boolean array $\text{shad}(\mathcal{B}, e) = (x_1, \ldots, x_r)$, such that

- $x_i = 1$ if and only if there is a call passing (downward) through e at round i of $\mathcal{B}$;
- $x_i = 0$ otherwise.

As a boolean array, $\text{shad}(\mathcal{B}, e) = (x_1, \ldots, x_r)$ can be viewed as the integer $\sum_{i=1}^{r} x_i 2^{r-i}$. This yields a natural ordering on the shadows.

Definition 2. Let $T = (V, E)$ be any directed tree, let $\mathcal{B}$ be a broadcast protocol from the root s of T, and let $e \in E$. $\mathcal{B}$ is said to be *lexicographically optimal on* e if $\text{shad}(\mathcal{B}, e) \leq \text{shad}(\mathcal{B}', e)$ for any broadcast protocol $\mathcal{B}'$ from s in T.

Note that a broadcast protocol $\mathcal{B}$ which is lexicographically optimal on e does not necessarily minimize the number of calls passing through e. However, lexical optimality is related to the minimum number of rounds as follows:

Lemma 1. *Let $T = (V, E)$ be a directed tree of root s. Let $u_1, \ldots, u_d$ be the d children of s, and let $e_i = (s, u_i)$, $i = 1, \ldots, d$. Under the all-port edge-disjoint paths mode, if a broadcast protocol from s to T is lexicographically optimal on all the e_i's, then it is optimal.*

Proof. Let $\mathcal{B}$ be a broadcast protocol from s to T, which is lexicographically optimal on all the e_i's, and let r be the completion time of $\mathcal{B}$. There is i such that $\text{shad}(\mathcal{B}, e_i) \geq 2^{r-1}$ because s necessarily gives a call at round 1. Let $\mathcal{B}'$ be an optimal broadcast protocol from s to T performing in r' rounds, $r' < r$. We have $\text{shad}(\mathcal{B}', e_i) \leq 2^{r'} - 1$ because $\text{shad}(\mathcal{B}', e_i)$ is on r' bits. Therefore, $\text{shad}(\mathcal{B}', e_i) < \text{shad}(\mathcal{B}, e_i)$, in contradiction with the lexicographic optimality of $\mathcal{B}$ on e_i. □

Actually, it is interesting to note that the all-port edge-disjoint paths mode allows us to construct broadcast protocols that are lexicographically optimal on every edge of the tree simultaneously, as stated in the following property whose proof can be found in [11].

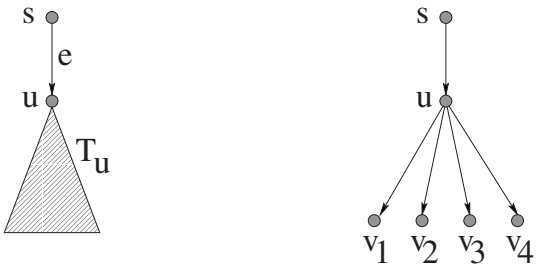

Fig. 1. Illustration of Lemma 2 and Remark 1.

Property 1. *Under the all-port edge-disjoint paths mode, for any directed tree T rooted at s, there exists an optimal broadcast protocol from s to T which is simultaneously lexicographically optimal on every edge of T.*

However, global lexical optimality cannot always be achieved for every variant of the line model. As a counterexample, consider the single-port constraint, and let T be the directed tree of three vertices, s, x, and y, and two arcs, $e_x = (s, x)$ and $e_y = (s, y)$. There are two possible optimal broadcast protocols from s to T: $\mathcal{B}_x$ consists in s calling x first, and then y; $\mathcal{B}_y$ consists in s calling y first, and then x. We have $\text{shad}(\mathcal{B}_x, e_y) = \text{shad}(\mathcal{B}_y, e_x) = (0, 1)$, but there is no broadcast protocol $\mathcal{B}$ satisfying $\text{shad}(\mathcal{B}, e_x) = \text{shad}(\mathcal{B}, e_y) = (0, 1)$. Nevertheless, the single-port vertex-disjoint paths mode satisfies the following property which is paramount for the purpose of this paper.

Lemma 2. *Let T be a directed tree rooted at s, and assume s has a unique child u (see Figure 1). Let $e = (s, u)$, and let $\mathcal{B}$ be a broadcast protocol from s to T under the single-port vertex-disjoint paths mode. For any integer $t > \text{shad}(\mathcal{B}, e)$, there is a broadcast protocol $\mathcal{B}_t$ from s to T such that $\text{shad}(\mathcal{B}_t, e) = t$. Moreover, given $\mathcal{B}$ and t, $\mathcal{B}_t$ can be constructed in $O(\log t)$ time.*

For instance, let $\text{shad}(\mathcal{B}, e) = (x_r, \ldots, x_1) = \sum_{i=1}^{r} x_i 2^{i-1}$ and $t = 2^r$. $\mathcal{B}_t$ is the broadcast protocol performing in $r + 1$ rounds as follows. At the first round, s calls u. All the remaining rounds are those of $\mathcal{B}$ except that calls are given by u instead of s which remains idle. Clearly, $\text{shad}(\mathcal{B}_t, e) = (1, 0, \ldots, 0) = 2^r$.

Proof. Let $t = (t_1, \ldots, t_{r'}) > \text{shad}(\mathcal{B}, e) = (x_1, \ldots, x_r)$. $\mathcal{B}_t$ performs as follows:

- If $r' > r$, then let $k = r' - r$. For every i, $1 \leq i \leq k$, if $t_i = 1$, then s calls u at round i of $\mathcal{B}_t$.
- If $r' = r$, then let k be the smallest index such that $t_k > x_k$. For every i, $1 \leq i < k$, $\mathcal{B}_t$ performs at round i as $\mathcal{B}$ does at the same round. At round k, s is idle in $\mathcal{B}$. In $\mathcal{B}_t$, s calls u if u is idle in $\mathcal{B}$, or calls v if u calls v in $\mathcal{B}$. In the latter case, u remains idle in $\mathcal{B}_t$.

The fact that u is informed at round k in $\mathcal{B}_t$ allows u to simulate the role of s in $\mathcal{B}$ during the

remaining rounds of $\mathcal{B}_t$. More precisely, let $x' = (x'_1, \ldots, x'_{r'}) = (0, \ldots, 0, x_1, \ldots, x_r)$ with $r' - r$ zeros inserted at the front, and let $i \in \{k+1, \ldots, r'\}$.

- If $t_i = x'_i$, then the calls of $\mathcal{B}_t$ performed at round i are the calls of $\mathcal{B}$ performed at round $i - (r' - r)$.
- If $t_i = 0$ and $x'_i = 1$, then the calls of $\mathcal{B}_t$ performed at round i are the calls of $\mathcal{B}$ performed at round $i - (r' - r)$, except that if s calls v in $\mathcal{B}$, then u calls v in $\mathcal{B}_t$, and s stays idle in $\mathcal{B}_t$. (This transformation is valid since u is necessarily idle in $\mathcal{B}$ because it is traversed by a call from s.)
- If $t_i = 1$ and $x'_i = 0$, then the calls of $\mathcal{B}_t$ performed at round i are the calls of $\mathcal{B}$ performed at round $i - (r' - r)$, except that if u is idle in $\mathcal{B}$, then s calls u in $\mathcal{B}_t$; otherwise, e.g., u calls v in $\mathcal{B}$, then s calls v in $\mathcal{B}_t$, and u stays idle in $\mathcal{B}_t$.

By construction, $\text{shad}(\mathcal{B}_t, e) = t$. Given $\mathcal{B}$ and t, the construction of $\mathcal{B}_t$ can be done in $O(\log t)$ phases by sequentially considering the $\lceil \log_2 t \rceil$ bits of the binary decomposition of t. At every phase, $\mathcal{B}_t$ is obtained from $\mathcal{B}$ by a constant number of operations. □

Remark 1. Using the same proof, it is easy to check that Lemma 2 holds for the all-port edge-disjoint paths mode as well. However, to understand the limit of our merging method, it is important to note that Lemma 2 does not hold in the single-port edge-disjoint paths mode. As a counterexample, let T be the directed tree of six nodes (see Figure 1): s, u, v_1, v_2, v_3, v_4, where s has a unique child u, and u has four children $v_i, i = 1, \ldots, 4$. Here is the list of calls of a broadcast protocol $\mathcal{B}$ from s to T performing in three rounds:

- Round 1: $s \to u$;
- Round 2: $s \to v_1$ and $u \to v_2$;
- Round 3: $s \to v_3$ and $u \to v_4$.

Let $e = (s, u)$ and let $t = 8 = (1000)_2$. We have $\text{shad}(\mathcal{B}, e) = (111)_2 = 7$, thus $t > \text{shad}(\mathcal{B}, e)$. However, one cannot broadcast from s to T in four rounds by giving a single call through e. Indeed, this call would have to be used by s to inform u (because u cannot be informed from the v_i's), and s would be idle the rest of the time. Node u would then need four additional rounds to inform the v_i's.

We conclude this section by some simple remarks. Let $T = (V, E)$ be a directed tree rooted at s. Let $\mathcal{B}$ be a broadcast protocol from s to T that is lexicographically optimal in $e \in E$, and let $\text{shad}(\mathcal{B}, e) = (x_1, \ldots, x_r)$. Let $\mathcal{B}'$ be another broadcast protocol from s to T, and let $\text{shad}(\mathcal{B}', e) = (y_1, \ldots, y_{r'})$, $r' \geq r$. Let $x' = (x'_1, \ldots, x'_{r'}) = (0, \ldots, 0, x_1, \ldots, x_r)$ with $r' - r$ zeros at the front. If there exists i such that $x'_i = 1$ and $y_i = 0$, then we say that x'_i *vanishes* in $\mathcal{B}'$. In that case, since $\mathcal{B}$ is lexicographically optimal in e, there exists $j < i$ such that $x'_j = 0$ and $y_j = 1$. Let k be the largest index smaller than i such that $x'_k < y_k$. The call x'_i of $\mathcal{B}$ is said to be *advanced* at round k in $\mathcal{B}'$. We summarize this discussion for further reference as follows:

Lemma 3. *In a directed tree, if a call through an edge e, given by a broadcast protocol $\mathcal{B}$ which is lexicographically optimal in e, vanishes in another broadcast protocol $\mathcal{B}'$, then this call must have been advanced in $\mathcal{B}'$.*

3. All-Port Broadcast in Directed Trees

This section is dedicated to the all-port edge-disjoint paths mode. It shows how to construct an optimal broadcast protocol in a tree by merging optimal protocols in subtrees. This merging technique will apply to the single-port constraint as well, but we will see that it cannot be done in such a simple manner. For the moment, we concentrate on the all-port constraint. Thus, we are given a tree T whose edges are directed away from its root s toward its leaves. We aim to compute an optimal broadcast protocol from s to T in the all-port edge-disjoint paths mode. Given a node v of T, we denote by T_v the subtree of T containing v and its descendents.

Lemma 4. *Let T be a directed n-node tree rooted at s. The broadcast from s to T takes at most $2\lceil \log_2 n \rceil$ rounds under the all-port edge-disjoint paths mode.*

Proof. The proof is by induction on $k = \lceil \log_2 n \rceil$. The result holds for $k = 1$. Assume it holds for $n \leq 2^k$, $k \geq 1$, and let $n \in \{2^k + 1, \ldots, 2^{k+1}\}$. Let x be the node of T such that $|T_x| \geq n/2$, and $|T_y| < n/2$ for every child y of x. (Such a property is satisfied by exactly one vertex x.) Now we consider the following broadcast protocol from s. At the first round s calls x. At the second round s is idle, and x calls all its children simultaneously. Then, by induction hypothesis, s can broadcast in $T \setminus T_x$ in at most $2k$ rounds, and every child y of x can broadcast in its subtree T_y in at most $2k$ rounds. Thus the whole protocol takes at most $2 + 2k = 2\lceil \log_2 n \rceil$ rounds. □

The following lemma is the kernel of our algorithm for the all-port constraint.

Lemma 5. *Let T be a directed n-node tree rooted at s, let $v \in V$, $v \neq s$, and let u be the parent of v in T. Let $T_1, \ldots, T_p$ be the p subtrees of T rooted at the p children $w_1, \ldots, w_p$ of v. Assume that, for any $i = 1, \ldots, p$, we know a broadcast protocol $\mathcal{B}_i$ from v to T_i which is lexicographically optimal on $e_i = (v, w_i)$. There is an $O(p \log n)$-time algorithm which returns a broadcast protocol from u to T_v which is lexicographically optimal on $e = (u, v)$.*

Proof. We merge the protocols $\mathcal{B}_i$'s. Let q_i be the number of rounds of $\mathcal{B}_i$, i.e., q_i is the length of the boolean array $\text{shad}(\mathcal{B}_i, e_i)$. Let $q = \max_{i=1,\ldots,p} q_i$, and let M be the $(p+1) \times q$ boolean matrix whose first row, say row 0, has a single 1-entry located at column q, and row i, $1 \leq i \leq p$, consists of $\text{shad}(\mathcal{B}_i, e_i)$ possibly complemented with 0-entries on the left if $q_i < q$. We label the columns of M from 1 to q, from left to right. From M, we construct a boolean matrix M' as follows. If every column of M contains at most one 1-entry, then $M' = M$. Otherwise, let c be the index of the leftmost column of M containing more than one 1-entry. (That is, c is the smallest index such that column c contains more than one 1-entry.) If there is no zero-column (that is, a column with 0-entries only) left of column c, then add a zero-column at the beginning of M, relabel the columns from 1 to $q+1$, and let $d = 1$. Otherwise, let $d < c$ be the index of the rightmost zero-column left to column c. Given d, move the 1-entry of row 0 from the last column to column d. The remaining entries are not modified, and the construction of M' is completed.

Let $\mathcal{B}$ be the broadcast protocol obtained from M' as follows. Let d be the position of the 1-entry on row 0 of M'. For every i, $i = 1, \ldots, d-1$, if v calls some node x at round i of some $\mathcal{B}_j$, then u calls x at round i of $\mathcal{B}$. At round d, u calls v. Then u stays idle for all the remaining rounds. All the other calls of $\mathcal{B}$ are those of the $\mathcal{B}_j$'s. Note that v may be involved in more than a single call simultaneously since two or more $\mathcal{B}_j$'s may simultaneously place a call from v after round d. Nevertheless, this is allowed by the all-port constraint.

The time required to construct $\mathcal{B}$ is the time it takes to construct M', that is, $O(pq)$. From Lemma 4, $q_i \leq 2\lceil \log_2 n \rceil$ for every i. Therefore, $\mathcal{B}$ is constructed in $O(p \log n)$ time.

It just remains to show that $\mathcal{B}$ is lexicographically optimal in e. If there has been no zero-column added to M, then $\text{shad}(\mathcal{B}, e)$ has q entries, and $\text{shad}(\mathcal{B}, e) = (x_1, \ldots, x_{d-1}, 1, 0, \ldots, 0)$ where, for $i < d$,

$$x_i = \sum_{j=1}^{p} \text{shad}(\mathcal{B}_j, e_j)_i \in \{0, 1\}.$$

If a zero-column has been added to M, then $\text{shad}(\mathcal{B}, e)$ has $q+1$ entries, and $\text{shad}(\mathcal{B}, e) = (1, 0, \ldots, 0)$. Assume for the purpose of contradiction that there exists a broadcast protocol $\mathcal{B}'$ from u to T_v such that $\text{shad}(\mathcal{B}', e) < \text{shad}(\mathcal{B}, e)$. Note that $\text{shad}(\mathcal{B}', e)$ then has exactly q entries. Let k be the round at which u calls v in $\mathcal{B}'$.

We consider two cases depending on whether or not M has a zero-column left of column c, i.e., whether $\text{shad}(\mathcal{B}, e)$ has q or $q+1$ entries. Assume first that $\text{shad}(\mathcal{B}, e)$ has $q+1$ entries. There are two subcases depending on when u calls v in $\mathcal{B}'$. If $k < c$, then, since there is no zero-column left of column c in M, some $\mathcal{B}_i$ is giving a call through e_i at round k. Therefore, from Lemma 3, this call has been advanced in $\mathcal{B}'$, say at round $k' < k$. In turn, since there is no zero-column left of column c in M, some $\mathcal{B}_{i'}$ is giving a call through $e_{i'}$ at round k'. Again, from Lemma 3, this call has hence been advanced in $\mathcal{B}'$, say at round $k'' < k' < k$. We construct in this way a decreasing sequence of indices. Since this sequence is bounded from below by 1, it necessarily ends, say at $k_0 \geq 1$. Again, since there is no zero-column left of column c in M, a call from some $\mathcal{B}_j$ vanishes at round k_0 in $\mathcal{B}'$. However, it is not advanced in $\mathcal{B}'$, in contradiction with Lemma 3. The case $k \geq c$ can be treated similarly, and the case where $\text{shad}(\mathcal{B}, e)$ has q entries as well (see the Appendix). One can check that all cases lead to a contradiction with Lemma 3. Therefore $\mathcal{B}$ is lexicographically optimal, which completes the proof. □

Theorem 1. *There is an $O(n \log n)$-time algorithm which, given any directed tree T rooted at s, returns an optimal broadcast protocol from s to T under the all-port edge-disjoint paths mode.*

Proof. The algorithm proceeds bottom-up, from the leaves toward the root. For each arc $e = (u, v)$ incoming to a leaf v, the optimal broadcast protocol $\mathcal{B}_u$ from u to T_v consists of a unique call from u to v. We get $\text{shad}(\mathcal{B}_u, e) = (1)$. Let $v \in V$, $v \neq s$, be an internal node, and let u be the parent of v in T. Let $T_1, \ldots, T_p$ be the p subtrees of T rooted at the p children $w_1, \ldots, w_p$ of v. Let $e_i = (v, w_i)$, $i = 1, \ldots, p$. Assume

that, for every $i = 1, \ldots, p$, we know a broadcast protocol $\mathcal{B}_i$, from v to T_i, that is lexicographically optimal in e_i. From Lemma 5, a broadcast protocol $\mathcal{B}_u$ from u to T_v, lexicographically optimal in $e = (u, v)$, can be constructed in $O(p \log n)$ time. At the root, let $S_1, \ldots, S_p$ be the p subtrees of T rooted at the p children $v_1, \ldots, v_p$ of s. Let $e_i = (s, v_i)$, $i = 1, \ldots, p$. Given p broadcast protocols $\mathcal{B}_i$, lexicographically optimal in e_i, respectively, merging these protocols at s takes a constant time since s can perform several calls simultaneously. The resulting broadcast protocol from s to T is lexicographically optimal in all incident edges of s. Therefore, thanks to Lemma 1, it is optimal.

The total time required by this bottom-up construction is $O(\sum_{v \in V} d(v) \cdot \log n)$ where $d(v)$ is the number of children of v in T. Therefore, the whole construction takes $O(n \log n)$ time. □

By similar techniques, one can extend the previous result to undirected trees. The proof is a bit more tricky because of the ability to place upward calls in undirected trees. This ability is captured by an extension of the definitions of shadow and lexical optimality. (In particular, shadows are arrays of elements in $\{-1, 0, 1\}$ rather than boolean arrays.) Based on this extension, the merging method applies similarly. We refer to [11] for more detail.

Theorem 2. *There is an $O(n \log n)$-time algorithm which, given any (undirected) tree T rooted at s, returns an optimal broadcast protocol from s to T under the all-port edge-disjoint paths mode.*

The next sections address the single-port model. As we will see, merging optimal protocols is not a task as easy as for the all-port model, and deriving the equivalent of Lemma 5 requires some more work.

4. The Contention-Free Matrix Problem

This section describes a polynomial-time algorithm for a matrix problem similar to the matrix problem solved in the proof of Lemma 5. The interest of this problem will appear clearly in the next section.

Definition 3. Given a $p \times q$ boolean matrix M, a *contention-free version* of M is a $p \times q'$ boolean matrix M', $q' \geq q$, such that:

- M' has at most one 1-entry per column, and
- every row r of M', viewed as the binary representation of an integer, is larger than the corresponding row r of M, $1 \leq r \leq p$.

For instance, for any $p \times q$ boolean matrix M, the $p \times (p + q)$ boolean matrix M' whose first p columns form the $p \times p$ identity matrix, and the last q columns form a zero-matrix, is a contention-free version of M. Such a solution M' may not be "minimum" even in terms of the number of columns. The following definition makes explicit the parameter that we want to optimize.

Definition 4. Let M be a $p \times q$ boolean matrix. The *shadow* of M, denoted by shad(M), is the boolean array of size q such that the ith entry of shad(M) is 1 if and only if there is at least one 1-entry in the ith column of M.

We look for contention-free versions of M that are optimal in the following sense.

Definition 5. Given a $p \times q$ boolean matrix M, a contention-free version M' of M is *minimal* if shad(M'), viewed as the binary representation of an integer, is minimum among the shadows of all the contention-free versions of M.

Note that there can be several minimal contention-free versions of a matrix, even up to a permutation of the rows. On the other hand, the shadow of the (possibly many) minimal contention-free versions of a matrix is unique. As an example, we consider the matrix

$$M = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

The reader can check that two minimal contention-free versions of M are

$$M_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad M_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

M_1 and M_2 are two different matrices, even up to a permutation of their rows. However, M_1 and M_2 both have a shadow equal to $62 = (111110)_2$.

Theorem 3. *There is an $O(q(p+q))$-time algorithm which computes a minimal contention-free version of any $p \times q$ boolean matrix.*

The remainder of this section is dedicated to the proof of this theorem. Let M be a $p \times q$ boolean matrix. We describe an algorithm which transforms M into a minimal contention-free version of M. Our algorithm proceeds via a sequence of elementary matrix operations of two types (columns are labeled from right to left[1]):

- insertion of a zero-column at the left of the current matrix, and
- shifting of an existing zero-column one position to the right, i.e., from its position t to position $t-1$ (in other words, performing an exchange between columns $t-1$ and t).

The shift operation has an important consequence on the 1-entries of the matrix. When a zero-column is shifted one position to the right, from position t to position $t-1$, the

[1] We label columns from right to left (as opposed to entries of shadows) in this section because the number of columns of the matrix changes during the execution of the algorithm, and new columns are added at the left of the matrix.

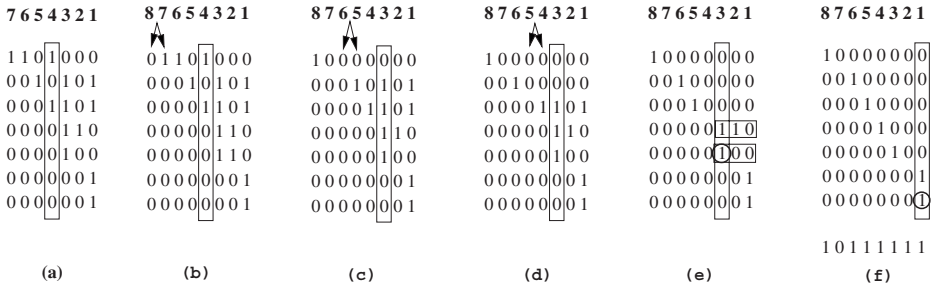

Fig. 2. An example of the execution of Algorithm `shift`.

entries of the matrix are modified according to the following rule:

`Switch-to-zero`: For every i, $1 \leq i \leq p$, if there is a 1-entry originally at column $t-1$ of row i, then, after the exchange of column $t-1$ with a zero-column at position t, all 1-entries of row i at position $t' < t-1$ are switched to 0.

Rule `switch-to-zero` is motivated by the fact that, for any k, $2^{k+1} > \sum_{i=0}^{k} a_i 2^i$ for any $a_i \in \{0, 1\}$, $i = 0, \ldots, k$. Therefore, any row modified according to Rule `switch-to-zero` is larger than the original, whatever its original entries are right of position t.

Our algorithm is called `shift`. Before describing it in detail, we consider an example of its execution on the matrix M of Figure 2(a). The columns of M are considered from left to right, until one meets a column containing two or more 1-entries. In Figure 2(a) there is a single 1-entry in each of the three leftmost columns of M. Contention occurs at column 4. Contentions are solved by the use of zero-columns placed on the left of the current column. The contending 1's are distributed over these zero-columns. At the current phase of the algorithm, there is no zero-column in M, and thus a zero-column is inserted on the left of the matrix, as shown on Figure 2(b). This zero-column is moved from left to right in order to be placed next to the current column. This is done via a sequence of column exchanges with application of the rule `switch-to-zero`. In our exemple, the two first columns are exchanged, and Rule `switch-to-zero` is applied. This application has a major consequence on the matrix: all 1-entries of the first row, except the leading 1-entry, are switched to 0. This creates a new zero-column, and one of the two contending 1's of column 4 becomes 0 (see Figure 2(c)). Hence, there is no more conflict at column 4, and Algorithm `shift` carries on checking the columns from left to right. Hence it now considers column 3. Four 1-entries are contending on this column. The effect of the shift of the rightmost zero-column on the left of column 3 is to delete one contending 1-entry (see Figure 2(d)). This zero-column is then shifted once more to the right. Again, it deletes one contending 1-entry (see Figure 2(e)). At this point, there is one zero-column and there are two contending 1's, so there is no need to insert a new zero-column. From column 3, row 4 is 110 whereas row 5 is 100, and $110 > 100$ in lexicographic order. Therefore the algorithm chooses to move the 1-entry of row 4 to the zero-column while the 1-entry of row 5 is left in place. At this point, we are left with the matrix on Figure 2(f) in which the last 1-entry of row 4 has been switched to 0.

Leaving the 1-entry of row 5 in place transforms the penultimate column of the matrix into a zero-column. Therefore, the conflict of column 1 can be solved easily by moving one of the two 1-entries on column 1 to column 2. The shadow of the resulting matrix is $(10111111)_2$, and we claim that it is minimum among all contention-free versions of the matrix of Figure 2(a).

We now describe Algorithm `shift` in detail (the algorithm itself is given in Figure 3).

Informal Description. Informally, Algorithm `shift` performs as follows. The q columns of the input matrix M are considered from left to right, that is, in decreasing order (Instruction 1). Problems occur when there are two or more 1-entries in the current column (Instruction 6). The goal of Algorithm `shift` is then to produce enough zero-columns on the left of the current column to solve this conflict, by distributing the contending 1's over these zero-columns. For this purpose, Algorithm `shift` tries to place zero-columns on the left of the current column by shifting existing zero-columns from their current position to their right, and by applying Rule `switch-to-zero` (Instruction 14). If there is no zero-column on the left of the current column, a zero-column is inserted at the leftmost position of M (Instruction 20). The algorithm repeats a move of one position to the right of the rightmost zero-column left to the current column. It is important to notice that it is the rightmost zero-column on the left of the current column which is considered. Choosing this column instead of any zero-column on the left of the current column has an important effect on the shadow of the resulting matrix because it minimizes the increment on the shadow resulting from moving a zero-column to the right. Note also that the rightmost zero-column may change because of the application of the Rule `switch-to-zero` which may create new zero-columns. In addition, Rule `switch-to-zero` may also reduce the number of contending 1-entries. The move-to-the-right operations are repeated until either all contenting 1-entries disappear, or there are enough zero-columns placed immediately on the left of the current column to solve all conflicts between the remaining 1-entries. In the former case, the algorithm carries on, and the next column is considerer. In the latter case, the remaining conflicts are solved as follows. The contending 1's of the current column are distributed over the zero-columns placed immediately on the left of this column, i.e., one 1-entry is placed on each zero-column, in an arbitrary order. Note that if after all possible shifts there is still not enough zero-columns to absorb the contending 1's, then some additional zero-columns are inserted. Moving these additional zero-columns from the leftmost column to a position immediately to the left of the current column has no effect on the shadow. Indeed, they are inserted after a zero-column has previously been inserted at the leftmost position of M, and moved rightward, to be placed left of the current column; this action has already caused all possible applications of Rule `switch-to-zero`. Therefore, the additional zero-columns can be placed directly immediately on the left of the current column (Instruction 26). The k contending 1's are distributed over $k-1$ zero-columns, one 1-entry remaining at its current position. The choice of the unique 1-entry which is not moved to a zero-column matters. Algorithm `shift` keeps in place the 1-entry which corresponds to the row with the *minimum lexicographic order*, starting from the current column (Instruction 28). Leaving the 1-entry of the smallest row, in lexicographic order,

Algorithm 1: computes a minimal contention-free version of a $p \times q$ boolean matrix M.

```
1  For i:=q to 1 do
   /* Columns are labeled from right to left */
2     C_i := current column;
3     If C_i is a zero-column then
4        Z := Z ∪ {C_i};
         /* Z denotes the set of zero-columns left of the current column */
5     Else
6        If there are more than a single 1-entry in C_i then
7           nb_1 := # 1's in C_i;
8           W := set of consecutive zero-columns immediately to the left of C_i;
9           not_yet_inserted := true;
10          While (nb_1 > |W| + 1) and (Z ≠ W or not_yet_inserted) do
            /* while there are not enough zero-columns immediately to the left of C_i */
11             If Z ≠ W then
               /* A zero-column can be moved rightward */
12                Z' := Z \ W;
13                c := rightmost zero-column in Z';
14                Shift c one column to the right, and apply Rule switch-to-zero;
15                Z := set of zero-columns left of C_i;
16                W := set of consecutive zero-columns immediately to the left of C_i;
17                nb_1 := # 1's in C_i;
18             EndIf
19             If (nb_1 > |W| + 1) and (W = Z) and (not_yet_inserted) then
               /* One needs to insert a new zero-column */
20                Insert a zero-column at the left of the matrix;
21                not_yet_inserted := false;
22                Z := set of zero-columns left of C_i;
23                W := set of consecutive zero-columns immediately to the left of C_i;
24             EndIf
25          EndWhile
26          If nb_1 > |W| + 1 then insert nb_1 − |W| − 1 zero-columns left of C_i;
            /* If there are not enough zero-columns to solve all contentions, */
            /* then additional zero-columns are inserted immediately to the left of C_i */
27          Truncate each row with a 1 in C_i in order to keep only entries to the right of C_i;
28          ℓ := index of the row with minimum lexicographic order among all truncated rows;
29          W' := nb_1 − 1 rightmost columns of W;
            /* The nb_1 1's are now distributed over the zero-columns of W */
30          Keep the 1-entry of row ℓ in place in C_i, and distribute the nb_1 − 1 other 1's of C_i over
            W';
31          Z := set of zero-columns left of the current column;
            /* The conflict at column C_i is now solved */
32       EndIf
33    EndIf
34 EndFor
```

Fig. 3. Algorithm `shift`.

in place has the effect of postponing other conflicts with this row as far as possible. Once the conflict has been solved by these operations, the next column is considered, and so on, until all columns have been considered.

The fact that Algorithm `shift` computes a minimal contention-free version of any $p \times q$ boolean matrix M is based on the following lemmas. For any matrix M, $\text{shad}^*(M)$ denotes the shadow of any minimal contention-free version of M. Note that

$\text{shad}^*(M) \geq \text{shad}(M)$, and that $\text{shad}^*(M) = \text{shad}(M)$ if and only if M has at most one 1-entry in each of its columns.

Lemma 6. *Let A_i and B_i be the i*th *row of two matrices A and B, respectively. If $A_i \leq B_i$ for every i, then* $\text{shad}^*(A) \leq \text{shad}^*(B)$.

Proof. Let B^* be a minimal contention-free version of B. For every i, $B_i^* \geq B_i \geq A_i$. Therefore, B^* is a contention-free version of A. Thus $\text{shad}^*(A) \leq \text{shad}(B^*) = \text{shad}^*(B)$. □

The next lemmas are related to the distribution of the contending 1-entries over adjacent zero-columns, as performed by algorithm `shift`. The following situation is therefore considered. Let A and B be two matrices of p rows, $p \geq 0$, and q_A and q_B columns, respectively. Let $X_i, Y_i, i = 1, \ldots, k$, be $2k$ one-dimensional boolean arrays, $k \geq 2$, where the X_i's are of size q_A and the Y_i's are of size q_B. Let

$$M = \begin{pmatrix} X_1 & 0 & \cdots & 0 & 1 & Y_1 \\ \vdots & \vdots & & \vdots & \vdots & \vdots \\ X_k & 0 & \cdots & 0 & 1 & Y_k \\ A & 0 & \cdots & 0 & 0 & B \end{pmatrix}$$

be a matrix of $p + k$ rows and $q_A + k + q_B$ columns (i.e., there are k zeros between A and B). Assume that there is at most one 1-entry in each of the q_A leftmost columns of M.

Lemma 7. *Let M^* be a minimal contention-free version of M. There is no zero-column in M^* between columns*[2] *$q_B + 1$ and $q_B + k$.*

Proof. For the purpose of contradiction, assume the reverse, and let c be the index of a zero-column in M^*, $q_B + 1 \leq c \leq q_B + k$. Let q^* be the number of columns of M^*. We consider R defined as the set of rows of M^* of the form

$$r^* = (x_{q^*}, \ldots, x_{q_B+k+1}, 0, \ldots, 0), \qquad x_i \in \{0, 1\},$$

and satisfying $r^* > r$ where r is the original row in M. In other words, R is the set of rows of M^* whose at least one 1-entry has been "moved" left to column $q_B + k$ in M^*. Since there is a zero-column between columns $q_B + 1$ and $q_B + k$ in M^*, and since the contention between the k 1-entries of column $q_B + 1$ must be solved, R is not empty. Let j be the index of the row of R whose rightmost 1-entry is the rightmost among all rows in R. Let c' be the column index of the rightmost 1-entry of row j. From M^*, we construct a matrix M' which is a copy of M^* except the following. Move the 1-entry of row j from position c' to position c (the index of the considered zero-column of M^* between

[2] Recall that the columns are labeled from right to left.

$$M^{(i)} = \left(\begin{array}{c|cccccccccc|cc}
X_1 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\
X_2 & 0 & 1 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots \\
X_{i-2} & 0 & 0 & \cdots & 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\
X_{i-1} & 0 & 0 & \cdots & 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\
\hline
X_i & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & Y_i \\
\hline
X_{i+1} & 0 & 0 & \cdots & 0 & 0 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 \\
X_{i+2} & 0 & 0 & \cdots & 0 & 0 & 0 & 1 & \cdots & 0 & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots \\
X_{k-1} & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 1 & 0 & 0 & 0 \\
X_k & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 0 & 0 \\
\hline
A & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & B
\end{array}\right)$$

Fig. 4. The matrix $M^{(i)}$.

columns $q_B + 1$ and $q_B + k$), and restore the original entries of this row between position c' and $q_B + k + 1$. The other entries of M' are those of M^*. M' is a contention-free version of M. Indeed, there is at most one 1-entry in each of the q_A leftmost columns of M, so M' has still at most one 1-entry on each column by the choice of row j. Moreover, row j of M' is still larger than the original row j of M. Since $\text{shad}(M') < \text{shad}(M^*)$, we get a contradiction. □

For $i = 1, \ldots, k$, let $M^{(i)}$ be the matrix described in Figure 4. $M^{(i)}$ is M where all but the ith contending 1's of column $q_B + 1$ have been distributed over the columns $q_B + 2, \ldots, q_B + k$, and Rule `switch-to-zero` has been applied. (Note that the fact the 1-entries form the identity matrix does not play any role, and this pattern has been chosen for easy presentation of the matrices only.)

Lemma 8. *Let i_0 be such that $Y_{i_0} \leq Y_i$ for every $i \neq i_0$. Then* $\text{shad}^*(M) = \text{shad}^*(M^{(i_0)})$.

Proof. From Lemma 6, $\text{shad}^*(M) \leq \text{shad}^*(M^{(i_0)})$. The reverse inequality is a bit more delicate to establish. Let M^* be a minimal contention-free version of M, and let q^* be the number of columns of M^*. We proceed in two cases:

Case 1. All the k 1-entries of column $q_B + 1$ of M have been moved to the left. Then, from Lemma 7, one can reorder entries of M^* between columns $q_B + 1$ and $q_B + k$ so that the resulting matrix N is a contention-free version of $M^{(i_0)}$. Indeed, the fact that there is no zero-column in M^* between columns $q_B + 1$ and $q_B + k$ make exchanges between the 1-entries on these columns possible, to fit with the placement of the 1's in $M^{(i_0)}$. Since $\text{shad}(N) = \text{shad}(M^*)$, we get $\text{shad}^*(M) \geq \text{shad}^*(M^{(i_0)})$.

Case 2. One of the k 1-entries of column $q_B + 1$ of M stands at the same position in M^*. Then let i be the row index of this entry.

If $i = i_0$, then, again, by reordering the positions of the $k - 1$ other 1-entries of M^* between columns $q_B + 1$ and $q_B + k$ solving the contentions at column $q_B + 1$ of M, M^*

can be transformed in a contention-free version N of $M^{(i_0)}$ with $\text{shad}(N) = \text{shad}(M^*)$. Hence $\text{shad}^*(M) \geq \text{shad}^*(M^{(i_0)})$.

If $i \neq i_0$, then let Y_i^* be the $q_B + k$ rightmost entries of row i in M^*, and let c be the column index of the rightmost 1-entry of row i_0 in M^*. We consider two subcases.

(a) If $c \leq q_B + k$, then let N be the matrix obtained from M^* by: (1) exchanging all the entries of row i with those of row i_0, from column 1 to column c; (2) reordering (thanks to Lemma 7) the positions of 1-entries of rows $1, \ldots, k$ of M^* between columns $q_B + 1$ and $q_B + k$ to make each of these rows larger than the corresponding row of $M^{(i_0)}$. We have $\text{shad}(N) = \text{shad}(M^*)$, and N is a contention-free version of $M^{(i_0)}$ because $Y_i^* \geq Y_i \geq Y_{i_0}$.

(b) If $c \geq q_B + k + 1$, then at least one of the 1-entries of the $p \times (q_A + k + q_B)$ matrix $A0B$ has been moved left to column $q_B + 1$ since, otherwise, there would be a zero-column in M^* between columns $q_B + 1$ and $q_B + k$, which is impossible by Lemma 7. Therefore, let R be the set of rows of the $p \times (q_A + k + q_B)$ matrix $A0B$ whose at least one 1-entry has been moved left to column $q_B + 1$. Let j be the index of the row of R whose rightmost 1-entry is the rightmost among all rows in R. Let c' be the column index of the rightmost 1-entry of row j. We have $c' \leq q_B + k$. Indeed, otherwise, there would be a zero-column between position $q_B + 1$ and $q_B + k$, in contradiction with Lemma 7. Then let N be the matrix obtained from M^* by: (1) exchanging all the entries of row i with those of row j, from column 1 to column c'; (2) reordering the positions of the 1-entries of rows $1, \ldots, k$ of M^* between columns $q_B + 1$ and $q_B + k$ to make each of these rows larger than the corresponding row of $M^{(i_0)}$. N is a contention-free version of $M^{(i_0)}$ because $1Y_i^* \geq 0B_j$. Moreover, $\text{shad}(N) = \text{shad}(M^*)$.

In both cases (a) and (b) we can construct a contention-free version N of $M^{(i_0)}$ such that $\text{shad}(N) = \text{shad}(M^*) = \text{shad}^*(M)$. Therefore $\text{shad}^*(M^{(i_0)}) \leq \text{shad}^*(M)$. □

Given two matrices A and B of the same number of rows p, and of q and q' columns, respectively, $A \mid B$ denotes the $p \times (q + q')$ matrix obtained by placing A and B next to each other, A on the left, and B on the right.

Lemma 9. *Let $M = A|x|B|y|C$ where A is a matrix of q_A columns, $q_A \geq 0$, with at most one 1-entry per column, x is a zero-column, B is a matrix of q_B columns, $q_B \geq 1$, with exactly one 1-entry per column, y is a column with at least two contending 1-entries, and C is an unspecified boolean matrix of q_C columns, $q_C \geq 0$. Let M' be the matrix resulting from M after an exchange between x and the leftmost column of B. We have* $\text{shad}^*(M) = \text{shad}^*(M')$.

Proof. From Lemma 6, $\text{shad}^*(M) \leq \text{shad}^*(M')$. The remainder of the proof is dedicated to the proof of the reverse inequality. Let M^* be a minimal contention-free version of M. Assume, without loss of generality, that the 1-entry of the leftmost column of B stands on row 1. Let M_i^* be the ith row of M^*. We consider two cases.

Case 1. $M_1^* \geq (A_1, 1, 0, \ldots, 0)$ where A_1 is the first row of A with $q_B + q_C + 1$ zeros. Then M^* is a contention-free version of M', and therefore $\text{shad}^*(M) \geq \text{shad}^*(M')$.

Case 2. $(A_1, 1, 0, \ldots, 0) > M_1^* \geq (A_1, 0, B_1, 0, C_1)$. Since at least one 1-entry of $B \mid y$ must be moved to the left of column $q_B + q_C + 1$ of M, let i be a row index of M^* such that (1) $M_i^* \geq (A_i, 1, 0, \ldots, 0)$, and (2) the rightmost 1-entry of M_i^* is the rightmost among rows satisfying inequality (1). Let c be the column-index of the rightmost 1-entry of M_i^*. Let N be the matrix obtained from M^* as follows:

1. The $q_B + 1 + q_C$ rightmost entries of row i are set to be equal to the $q_B + q_C + 1$ rightmost entries of row 1.
2. The entries of row i from column $q_B + q_C + 1$ to column c are restored as they were originally in M.
3. The entry at column c of row 1 is set to 1.
4. The entries of row 1 right to column c are set to 0.
5. All other entries are those of M^*.

Since M^* and A are matrices with at most one 1-entry per column, the resulting matrix N satisfies the same property by the choice of row i. Moreover, we have $(B_1, y_1, C_1) > (B_i, y_i, C_i)$ because, by hypothesis, the 1-entry of the leftmost column of B stands on row 1. Therefore, N is a contention-free version of M'. $\text{shad}(N) = \text{shad}(M^*)$, therefore $\text{shad}^*(M) \geq \text{shad}^*(M')$. □

Now, we know enough to prove Theorem 3.

Proof of Theorem 3. Algorithm `shift` constructs a finite sequence of matrices $M_0 = M, M_1, \ldots, M_k$, such that M_i is obtained from M_{i-1} either by shifting a zero-column to the right, or by distributing 1-entries over zero-columns. Lemma 8 shows that the distribution of the 1's over the zero-column preserves $\text{shad}^*(M)$. Similarly, Lemma 9 shows that the shift of a zero-column also preserves $\text{shad}^*(M)$. Therefore, $\text{shad}^*(M_i) = \text{shad}^*(M_{i-1})$, that is, $\text{shad}^*(M) = \text{shad}^*(M_k)$. Since M_k is a contention-free version of M, we get that $\text{shad}^*(M) = \text{shad}(M_k)$.

It just remains to compute the time-complexity of Algorithm `shift`. The *for*-loop is executed q times. However, Instruction 5 (the else block) is not performed more than p times in total. Indeed, there are p rows, and solving a contention between 1-entries creates at least one row whose all entries are null on the right of the current position. Let i be an index of the *for*-loop for which there is a contention. Again, there are at most p such indices. Let k_i be the number of contending 1-entries. All instructions before the *while*-loop do not require more than $O(p + q)$ time units. The *while*-loop is executed at most qk_i times because each execution of the loop corresponds to a right-shift of a zero-column, and one cannot move a zero-column more than q times to the right, this for each of the k_i 1-entries. Actually, one can slightly modify the algorithm so that there are no more than q right-shifts in total. Indeed, when shifting the zero-columns to the right, one can jump columns that were previously exchanged with a zero-column because Rule `switch-to-zero` was already applied for these columns. Therefore, Rule `switch-to-zero` is not applied more than q times. Application of Rule `switch-to-zero` has a cost of $O(q)$ since at most one row is updated after a right-shift. All other instructions inside the *while*-loop have a cost of $O(p+q)$. Instruction 28 has a cost of $O(q\ k_i)$, the same as Instruction 30. Therefore, the total time-complexity of Algorithm `shift` is $O(q(q + p) + q^2 + \sum_i q\ k_i)$. We have $\sum_i k_i \leq 2p$ because

solving contending 1-entries at a column c removes the 1-entries on the right of column c in all but one of the contending rows. Therefore, the time-complexity of Algorithm `shift` is $O(q(q+p))$. □

5. Single-Port Broadcast in Directed Trees

In this section we show how to apply Theorem 3 to derive optimal broadcast protocols in trees under the single-port constraint. We first give an illustration of the use of contention-free matrices for the construction of broadcast protocols.

Let T be a directed tree rooted at s, and let $T_1, \ldots, T_p$ be the p subtrees of T rooted at the p children $u_1, \ldots, u_p$ of s. Assume that, for every $i = 1, \ldots, p$, we know a broadcast protocol $\mathcal{B}_i$ from s in T_i which is lexicographically optimal on $e_i = (s, u_i)$. Let q_i be the number of entries in $\text{shad}(\mathcal{B}_i, e_i)$. Let M be the $p \times q$ boolean matrix whose p rows are the p shadows $\text{shad}(\mathcal{B}_i, e_i)$, possibly complemented with zero-entries on the left if $q > q_i$. Let M^* be a $p \times q^*$ minimal contention-free version of M obtained by application of Algorithm `shift` on M.

Let $\mathcal{B}$ be the broadcast protocol from s in T derived from M^* as follows. Columns are labeled from left to right, from 1 to q^*. If there is a 1-entry on column i, say on row $j \geq 1$, then s gives a call inside T_j at round i, otherwise s stays idle. The destinations of the calls performed by s is not specified but, since each row of M^* is larger than the corresponding row of M, we get from Lemma 2 that every row j of M^* describes a broadcast protocol $\mathcal{C}_j$ in T_j.

$\mathcal{B}$ is optimal. Indeed, assume for the purpose of contradiction that there is a broadcast protocol $\mathcal{B}'$ from s to T performing in $q' < q^*$ rounds. Then let M' be the $p \times q'$ bolean matrix whose ith row is $\text{shad}(B', e_i)$. The single-port constraint implies that M' is a contention-free version of M, a contradiction with the fact that a minimal contention-free version of M has $q^* > q'$ columns.

Therefore, we have:

Lemma 10. *Given lexicographically optimal broadcast protocols $\mathcal{B}_i$ from s in T_i, there exists an $O(pn)$-time algorithm which returns the broadcast protocol $\mathcal{B}$ from s in T.*

Proof. Lemma 2 specifies that $\mathcal{C}_j$ can be obtained from $\mathcal{B}_j$ and the jth row of M^* in $O(q)$ time. Thus the construction of $\mathcal{B}$ requires $O(pq)$ time, once the matrix M^* has been computed. From Theorem 3, it takes $O(p(p+q))$ time to compute M^*. So the whole construction of $\mathcal{B}$ takes $O(p(p+q)) = O(pn)$ time since $q \leq n$. □

To construct the whole protocol, we apply a bottom-up construction consisting of merging protocols of intermediate levels, just as for the all-port edge-disjoint paths mode. The merging procedure at intermediate levels is slightly more complex than the merging at the root described above.

Lemma 11. *Let T be a directed tree rooted at s, let $v \in V$, $v \neq s$, and let u be the parent of v. Let $T_1, \ldots, T_p$ be the p subtrees of T rooted at the p children $w_1, \ldots, w_p$*

of v. Assume that, for every $i = 1, \ldots, p$, we know a broadcast protocol $\mathcal{B}_i$ from v in T_i which is lexicographically optimal on $e_i = (v, w_i)$. There exists an $O(pn)$-time algorithm which returns a broadcast protocol from u in T_v that is lexicographically optimal on $e = (u, v)$.

Proof. As in the proof of Lemma 10, let M be the $p \times q$ boolean matrix whose p rows are the p shadows $\text{shad}(\mathcal{B}_i, e_i)$, possibly complemented with zero-entries on the left if q is larger than the number q_i of entries in $\text{shad}(\mathcal{B}_i, e_i)$. Let M^* be a $p \times q^*$ minimal contention-free version of M obtained by application of Algorithm `shift` on M.

Let $\bar{M}$ be the $(p+1) \times q^*$ boolean matrix obtained from M^* by adding a new row, say row 0. This row has all its entries set to 0, but one. The single 1-entry of row 0 of $\bar{M}$ is placed at the column-index of the rightmost zero-column of M^* (if any). If M^* has no zero-column, then a zero-column is added at the left of M^*, and the 1-entry of row 0 of $\bar{M}$ is placed on this column.

Let $\mathcal{B}$ be the broadcast protocol from u in T_v derived from $\bar{M}$ as follows. Columns are labeled from left to right, from 1 to q^* or $q^* + 1$. Let c be the index of the column containing the 1-entry on row 0 of $\bar{M}$. Let $i < c$. If there is a 1-entry on column i, say on row $j \geq 1$, then u gives a call inside T_j at round i, otherwise u stays idle. At round c, u calls v. During the remaining calls, u stays idle. Let $i > c$. If there is a 1-entry on column i, say on row $j \geq 1$, then v gives a call inside T_j at round i. The destinations of the calls performed from u or v at rounds $i \neq c$ are not specified here. However, since each row of $\bar{M}$ is larger than the corresponding row of M, we get from Lemma 2 that every row $j \geq 1$ of $\bar{M}$ describes a broadcast protocol $\mathcal{B}'_j$ in T_j. $\mathcal{B}'_j$ can be obtained from $\mathcal{B}_j$ and the jth row of $\bar{M}$ in $O(q)$ time. Thus the construction of $\mathcal{B}$ requires $O(pq)$ time, once the matrix $\bar{M}$ has been computed. From Theorem 3, it takes $O(p(p+q))$ time to compute M^*, and $\bar{M}$ just requires $O(pq)$ additional time units to be constructed. So the whole construction of $\mathcal{B}$ takes $O(p(p+q)) = O(pn)$ time since $q \leq n$.

It remains to show that $\mathcal{B}$ is lexicographically optimal on e. Recall that c is the index of the column containing the 1-entry of row 0 of $\bar{M}$. If

$$\text{shad}(M^*) = (x_1, \ldots, x_{c-1}, 0, 1, \ldots, 1),$$

where $x_i \in \{0, 1\}$ for every $i = 1, \ldots, c-1$, then

$$\text{shad}(\mathcal{B}, e) = (x_1, \ldots, x_{c-1}, 1, 0, \ldots, 0).$$

If

$$\text{shad}(M^*) = (1, \ldots, 1)$$

with q^* ones, then

$$\text{shad}(\mathcal{B}, e) = (1, 0, \ldots, 0)$$

with q^* zeros. Assume for the purpose of contradiction that there is a broadcast protocol $\mathcal{B}'$ such that $\text{shad}(\mathcal{B}', e) < \text{shad}(\mathcal{B}, e)$. Then let $\bar{M}'$ be the boolean matrix of $p+1$ rows

such that there is a 1-entry at row $i \in \{1, \ldots, p\}$ and column j of $\bar{M}'$ if and only if u or v gives a call to a node of T_i at round j of $\mathcal{B}'$. Moreover, there is a 1-entry on row 0 and column j of $\bar{M}'$ if and only if u calls v at round j of $\mathcal{B}'$. Let M' be the boolean matrix obtained from $\bar{M}'$ by removing row 0. M' is a contention-free version of M because (1) $\text{shad}(\mathcal{B}', e_i) \geq \text{shad}(\mathcal{B}_i, e_i)$ for every i (since the $\mathcal{B}_i$'s are lexicographically optimal), and (2) there is at most one 1-entry per column of M' (since $\mathcal{B}'$ satisfies the single-port constraint). Let

$$\text{shad}(M') = (y_1, \ldots, y_{d-1}, 0, z_1, \ldots, z_r),$$

where $r \geq 0$, $y_i \in \{0, 1\}$, and $z_i \in \{0, 1\}$ for all i, and d is the round at which u calls v in $\mathcal{B}'$. We have

$$\text{shad}(\mathcal{B}', e) = (y_1, \ldots, y_{d-1}, 1, 0, \ldots, 0).$$

We show that $\text{shad}(\mathcal{B}', e) < \text{shad}(\mathcal{B}, e)$ and $\text{shad}(M^*) \leq \text{shad}(M')$ are in contradiction. This is clear if $\bar{M}$ has $q^* + 1$ columns, i.e., $\text{shad}(M^*) = (1, \ldots, 1)$. If $\bar{M}$ has q^* columns, then we consider three cases according to the relative values of c and d:

- If $c > d$, then $\text{shad}(\mathcal{B}', e) < \text{shad}(\mathcal{B}, e)$ and $\text{shad}(M^*) \leq \text{shad}(M')$ imply $(x_1, \ldots, x_{d-1}) = (y_1, \ldots, y_{d-1})$. Then $\text{shad}(M^*) \leq \text{shad}(M')$ implies $x_d = 0$. On the other hand, $\text{shad}(\mathcal{B}', e) < \text{shad}(\mathcal{B}, e)$ implies $x_d = 1$, a contradiction.
- If $c = d$, then $\text{shad}(\mathcal{B}', e) < \text{shad}(\mathcal{B}, e)$ implies that $(y_1, \ldots, y_{d-1}) < (x_1, \ldots, x_{d-1})$. On the other hand, $\text{shad}(M^*) \leq \text{shad}(M')$ implies that $(x_1, \ldots, x_{d-1}) \leq (y_1, \ldots, y_{d-1})$, a contradiction.
- If $c < d$, then $\text{shad}(\mathcal{B}', e) < \text{shad}(\mathcal{B}, e)$ and $\text{shad}(M^*) \leq \text{shad}(M')$ imply $(x_1, \ldots, x_{c-1}) = (y_1, \ldots, y_{c-1})$. Then $\text{shad}(\mathcal{B}', e) < \text{shad}(\mathcal{B}, e)$ implies $y_c = 0$. However, then $\text{shad}(M^*) > \text{shad}(M')$ because $y_d = 0$, a contradiction.

Every case yields a contradiction. Therefore, $\mathcal{B}$ is lexicographically optimal on e. □

Theorem 4. *There is an $O(n^2)$-time algorithm which, given any directed tree T rooted at s, returns an optimal broadcast protocol from s to T under the single-port vertex-disjoint paths mode.*

Proof. Same as the proof of Theorem 1. Lemma 10 replaces Lemma 1, and Lemma 11 replaces Lemma 5. The time complexity changes because merging protocols under the single-port vertex-disjoint paths mode is more costly than under the all-port edge-disjoint paths mode. The total time required by the bottom-up construction is $O(\sum_{v \in V} d(v) \cdot n) = O(n^2)$. □

Corollary 1. *There exists an $O(n^2)$-time 2-approximation algorithm for the broadcast problem in directed trees under the single-port edge-disjoint paths mode.*

Proof. Let T be a directed tree rooted at s. Let t_e and t_v be respectively the broadcast time from s to T under the single-port edge-disjoint paths mode, and the broadcast time

from s to T under the single-port vertex-disjoint paths mode. We have

$$t_e \leq t_v \leq 2t_e. \tag{1}$$

The first inequality $t_e \leq t_v$ is straightforward since pairwise vertex-disjoint paths are pairwise edge-disjoint. The second inequality comes from the fact that, in a directed tree, any set S of pairwise edge-disjoint paths can be decomposed into two subsets S' and S'' such that two paths belonging to the same subset are vertex-disjoint. Indeed, let G be the graph whose vertices are the paths in S, and there is an edge between two paths if they share any vertex of T. Since T is a tree, there is no cycle in G, and thus G is a forest. Therefore, G is bipartite, and its vertices can be colored with two colors. This completes the proof of (1). Since Theorem 4 says that an optimal broadcast algorithm under the single-port vertex-disjoint paths mode can be computed in $O(n^2)$-time, the result follows. □

6. Extensions and Concluding Remarks

All algorithms presented in this paper trivially extend to the multicast problem in which we are given a graph (or a digraph) $G = (V, E)$, a source node $s \in V$, a destination set $D \subseteq V$, and we have to compute an optimal protocol ensuring the dissemination of information from s to D is G. We refer the reader to [11] for the formal proofs. Actually, the results hold even under the *restricted regimen* of the considered communication model, in which we are given a set $\mathcal{F}$ of "forbidden nodes," i.e., nodes that cannot be used to relay messages. (These nodes can be traversed by calls, but cannot be used to receive a message at a given round for the purpose of forwarding this message a further round.) Again, we refer the reader to [11] for more details.

As far as trees are concerned, two problems remain: broadcasting in directed trees under the single-port edge-disjoint paths mode, and broadcasting in trees under the single-port vertex-disjoint paths mode. In these two cases, only $O(1)$-approximation algorithms are known (see [5], [6], [22], and Corollary 1). The remark after Lemma 2 explains why our technique does not apply to the single-port edge-disjoint paths mode since there is no one-to-one correspondence between binary words and shadows of broadcast protocols. The failure of our approach for the single-port vertex-disjoint paths mode in undirected trees is due to the difficulty of merging shadows with entries in $\{-1, 0, 1\}$ in the single-port model.

Appendix. Proof of Lemma 5 (continued)

Assume that $\text{shad}(\mathcal{B}, e)$ has $q + 1$ entries, and $k \geq c$. The contending 1's of column c imply that one of these calls must be advanced in $\mathcal{B}'$ at round $k' < c \leq k$. Applying the same reasoning as for the case $k < c$, we construct again a decreasing sequence of indices, yielding a contradiction with Lemma 3.

If $\text{shad}(\mathcal{B}, e)$ has q entries, then let $\text{shad}(\mathcal{B}', e) = (y_1, \ldots, y_q)$, and let ℓ be the smallest index such that $y_\ell < x_\ell$. Note that $\ell \leq d$ because $x_i = 0$ for every $i > d$.

Moreover, $\ell \neq k$ because $y_k = 1$ whereas $y_\ell = 0$. We consider two subcases depending on the relative values of k and ℓ:

(a) $k < \ell$. Since $y_i = x_i$ for every $i < \ell$, in particular $x_k = y_k = 1$. Thus some $\mathcal{B}_i$ is giving a call through e_i at round k. From Lemma 3, this call must be advanced in $\mathcal{B}'$, say at round $k' < k$. Again, $x_{k'} = y_{k'} = 1$, and by repeating the same argument, we get a contradiction with Lemma 3.

(b) $\ell < k$. Two subcases:

(i) $k \leq d$. Hence we have $\ell < d$, and thus, since $x_\ell = 1$, some $\mathcal{B}_i$ is giving a call through e_i at round ℓ. Since $y_\ell = 0$ and v is not yet informed at the ℓth round of $\mathcal{B}'$ (as we assumed $\ell < k$), Lemma 3 implies that this call must be advanced in $\mathcal{B}'$, say at round $k' < \ell < k$. Applying the reasoning of (a), we get again a contradiction with Lemma 3.

(ii) $k > d$. Two subcases:

- If $d < k < c$, then since there is exactly one 1-entry on every column of M between column d and column c, some $\mathcal{B}_i$ is giving a call through e_i at round k. From Lemma 3, this call must be advanced in $\mathcal{B}'$ at round $k' < k$. The repetition of the same argument on k' depends on its relative value with ℓ and d. Anyway, it will fall in one of the previous cases, including the current one, leading to the same contradiction with Lemma 3.
- If $c \leq k$, then the contending 1's of column c imply that one of these calls must be advanced in $\mathcal{B}'$ at round $k' < c \leq k$. We hence fall in one of the previous cases, and we get a contradiction with Lemma 3. □

References

[1] T. Ballardie, P. Francis, and J. Crowcroft. Core based tree (CBT): an architecture for scalable inter-domain multicast routing. In *Proc. ACM SIGCOMM*, pages 85–95, 1993.

[2] A. Bar-Noy, S. Guha, J. Naor, and B. Schieber. Multicasting in heterogeneous networks. In *Proc.* 30*th ACM Symposium on Theory of Computing* (*STOC* '98), pages 448–453, 1998.

[3] B. Beauquier, J.-C. Bermond, L. Gargano, P. Hell, S. Pérennes, and U. Vaccaro. Graph problems arising from wavelength-routing in all-optical networks. In *Proc.* 2*nd IPPS Workshop on Optics and Computer Science* (*WOCS* '97), 1997.

[4] J.-C. Bermond, A. Bonnecaze, T. Kodate, S. Pérennes, and P. Sole. Broadcasting in hypercubes under the circuit-switched model. In *Proc. IEEE International Parallel and Distributed Processing Symposium* (*IPDPS*), 2000.

[5] B. Birchler, A.-H. Esfahanian, and E. Torng. Toward a general theory of unicast-based multicast communication. In *Proc.* 21*st Workshop on Graph-Theoretic Concepts in Computer Science* (*WG* '95), pages 237–251. Volume 1017 of LNCS, Springer-Verlag, Berlin, 1995.

[6] B. Birchler, A.-H. Esfahanian, and E. Torng. Information dissemination in restricted routing networks. In *Proc. International Symposium on Combinatorics and Applications*, pages 33–44, 1996.

[7] C.-T. Chou and I. Gopal. Linear broadcast routing. *Journal of Algorithms*, 10:490–517, 1989.

[8] I. Cidon, I. Gopal, and S. Kutten. New models and algorithms for future networks. *IEEE Transactions on Information Theory*, IT-41:769–780, 1995.

[9] J. Cohen. Broadcasting, multicasting and gossiping in trees under the all-port line model. In *Proc.* 10*th ACM Symposium on Parallel Algorithms and Architectures* (*SPAA* '98), pages 164–171, 1998.

[10] J. Cohen. Communications multipoints dans le modèle commuté. Ph.D. thesis, LRI, Université Paris-Sud, 1999.
[11] J. Cohen and P. Fraigniaud. Broadcasting and multicasting in trees. Technical Report LRI-1265, Laboratoire de Recherche en Informatique, Université Paris-Sud, 2000. (See http://www.lri.fr/~pierre.)
[12] J. Cohen, P. Fraigniaud, J.-C. Konig, and A. Raspaud. Optimized broadcasting and multicasting protocols in cut-through routed networks. *IEEE Transaction on Parallel and Distributed Systems*, 9(8):788–802, 1998.
[13] J. Cohen, P. Fraigniaud, and M. Mitjana. Minimal contention-free matrices with application to multicasting. In *Proc. Workshop on Robust Communication Networks*: *Interconnection and Survivability*, pages 17–33. Volume 53 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, Providence, RI, 1998.
[14] J. Cohen, P. Fraigniaud, and M. Mitjana. Scheduling calls for multicasting in tree-networks (short paper). In *Proc. 10th ACM–SIAM Symposium on Discrete Algorithms* (*SODA* '99), pages 881–882, 1999.
[15] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C-G. Liu, and L. Wei. An architecture for wide-area multicast routing. In *Proc. SIGCOMM*, pages 126–135. ACM Press, New York, 1994.
[16] C. Diot, W. Dabbous, and J. Crowcroft. Multipoint communication: a survey of protocols, functions, and mechanisms. *IEEE Journal on Selected Areas in Communications*, 15(3):277–290, 1997.
[17] J. Duato, S. Yalamanchili, and L. Ni. *Interconnection Networks. An Engineering Approach*. IEEE Computer Society Press, Los Alamitos, CA, 1997.
[18] T. Eilam, M. Flammini, and S. Zaks. A complete characterization of path layout construction problem for ATM networks with given hop count and load. In *Proc. 24th International Colloquium on Automata, Languages and Programming* (*ICALP*), pages 527–537. Volume 1256 of LNCS, Springer-Verlag, Berlin, 1997.
[19] M. Elkin and G. Kortsarz. Combinatorial logarithmic approximation algorithm for the dirceted telephone broadcast problem. In *Proc. 34th Annual ACM Symposium on Theory of Computing* (*STOC*), pages 438–447, 2002.
[20] A. Farley. Minimum-time line broadcast networks. *Networks*, 10:59–70, 1980.
[21] A. Farley and A. Proskurowski. Broadcasting in trees with multiple originators. *SIAM Journal on Algebraic and Discrete Methods*, 2(4):381–386, 1981.
[22] P. Fraigniaud. Approximation algorithms for minimum-time broadcast under the vertex-disjoint paths mode. In *Proc. 9th Annual European Symposium on Algorithms* (*ESA* '01), pages 440–451. Volume 2161 of LNCS, Springer-Verlag, Berlin, 2001.
[23] P. Fraigniaud. Minimum-time broadcast under edge-disjoint paths modes. In *Proc. 2nd International Conference on Fun with Algorithms* (*FUN* '01), pages 133–148. Carleton Scientific, Ottawa, 2001.
[24] P. Fraigniaud and E. Lazard. Methods and problems of communication in usual networks. *Discrete Applied Mathematics*, 53:79–133, 1994.
[25] P. Fraigniaud and S. Vial. Approximation algorithms for broadcasting and gossiping. *Journal of Parallel and Distributed Computing*, 43:47–55, 1997.
[26] H. Harutyunyan. Doctoral Thesis (in Russian), Institute of Informatics of the Armenian Academy of Sciences, Yerevan, Armenia.
[27] S. M. Hedetniemi, S. T. Hedetniemi, and A. Liestman. A survey of gossiping and broadcasting in communication networks. *Networks*, 18:319–349, 1986.
[28] J. Hromković, R. Klasing, B. Monien, and R. Peine. Dissemination of information in interconnection networks (broadcasting and gossiping). In Ding-Zhu Du and D. F. Hsu, editors, *Combinatorial Network Theory*, pages 125–212. Kluwer, Dordrecht, 1995.
[29] J. Hromković, R. Klasing, and E. Stohr. Dissemination of information in vertex-disjoint paths mode. *Computer and Artificial Intelligence*, 15(4):295–318, 1996.
[30] J. Hromković, R. Klasing, E. Stohr, and H. Wagener. Gossiping in vertex-disjoint paths mode in d-dimensional grids and planar graphs. *Information and Computation*, 123(1):17–28, 1995.
[31] J. Hromković, R. Klasing, W. Unger, and H. Wagener. Optimal algorithms for broadcast and gossip in the edge-disjoint paths mode. *Information and Computation*, 133(1):1–33, 1997.
[32] G. Kortsarz and D. Peleg. Approximation algorithms for minimum time broadcast. *SIAM Journal on Discrete Methods*, 8:401–427, 1995.
[33] R. Labahn. Extremal broadcasting problems. *Discrete Applied Mathematics*, 23:139–155, 1989.

[34] M. Middendorf. Minimum broadcast time is NP-complete for 3-regular planar graphs and deadline 2. *Information Processing Letters*, 46:281–287, 1993.

[35] W. Mostafa and M. Singhal. A taxonomy of multicast protocols for internet applications. *Computer Communications*, 20:1448–1457, 1998.

[36] J. Peters and M. Syska. Circuit-switched broadcasting in torus networks. *IEEE Transactions on Parallel and Distributed Systems*, 7(3):246–255, 1996.

[37] A. Proskurowski. Minimum broadcast trees. *IEEE Transactions on Computers*, C-30:363–366, 1981.

[38] R. Ravi. Rapid rumor ramification: approximating the minimum broadcast time. In *Proc*. 35*th IEEE Symposium on Foundations of Computer Science*, pages 202–213, 1994.

[39] P. Scheuermann and G. Wu. Heuristic algorithms for broadcasting in point-to-point computer networks. *IEEE Transactions on Computers*, C-33(9):804–811, 1984.

[40] P. Slater, E Cockayne, and S. Hedetniemi. Information dissemination in trees. *SIAM Journal on Computing*, 10(4):692–701, 1981.

Received December 24, 2001. *Online publication September* 20, 2002.