

# Deterministic M2M Multicast in Radio Networks

## (Extended Abstract)

Leszek Gąsieniec<sup>1\*</sup>, Evangelos Kranakis<sup>2\*\*</sup>, Andrzej Pelc<sup>3\*\*\*</sup>, and Qin Xin<sup>1</sup>

<sup>1</sup> Department of Computer Science, University of Liverpool, Liverpool L69 7ZF, UK,  
{leszek,qinxin}@csc.liv.ac.uk

<sup>2</sup> School of Computer Science, Carleton University, Ottawa, Ontario, K1S 5B6, Canada,  
kranakis@scs.carleton.ca

<sup>3</sup> Dép. d'informatique, Université du Québec en Outaouais, Hull, Québec, J8X 3X7, Canada,  
andrzej.pelc@uqo.ca

**Abstract.** We study the problem of exchanging messages within a fixed group of  $k$  nodes, in an  $n$ -node multi-hop radio network, also known as the problem of Multipoint-to-Multipoint (M2M) multicasting. While the radio network topology is known to all nodes, we assume that the participating nodes are not aware of each other's positions. We give a new fully distributed deterministic algorithm for the M2M multicasting problem, and analyze its complexity. We show that if the maximum distance between any two out of  $k$  participants is  $d$  then this local information exchange problem can be solved in time  $O(d \log^2 n + k \log^3 n)$ . Hence our algorithm is linear in the size of the subnetwork induced by the participating nodes and only polylogarithmic in the size of the entire radio network.

## 1 Introduction

Next generation wireless networks are expected to support group communication applications (such as distance learning, video conferencing, disaster recovery and distributed collaborative computing). In such applications, any of the nodes of a well-defined group may be required to send messages to all other nodes in the group. The problem of exchanging messages within a fixed group of nodes in a multi-hop network is called *M2M (multipoint-to-multipoint) multicasting*.

*Broadcasting* and *gossiping* are two classical problems of information dissemination in computer networks. In the broadcasting problem, we want to distribute a message from a distinguished *source* node to all other nodes in the network. In the gossiping problem, each node  $v$  in the network initially holds a message  $m_v$ , and we wish to distribute all messages  $m_v$  to all nodes in the network. In both problems, one of the main efficiency criteria is the time needed to complete the given communication task. M2M multicasting is a natural generalization of gossiping, in which information exchange concerns not all nodes of the network but only a subset of all nodes, called *participants*.

---

\* Research supported in part by the Royal Academy of Engineering.

\*\* Research supported in part by MITACS, and NSERC grants.

\*\*\* Research supported in part by NSERC grant and the Research Chair in Distributed Computing of the Université du Québec en Outaouais.

A radio network is a collection of stations, equipped with capabilities of transmitting and receiving messages. Stations will be referred to as *nodes* of the network. The network is modeled as an  $n$ -node undirected connected graph  $G = (V, E)$  on the set of these nodes. Each node has a unique label drawn from set  $[N] = \{0, 1, \dots, N - 1\}$  of integers, where  $N$  is bounded by some polynomial in  $n$ . An edge  $e$  between two nodes means that the transmitter of one end of  $e$  can reach the other end. Nodes send messages in synchronous *steps* (time slots). In every step every node acts either as a *transmitter* or as a *receiver*. A node acting as a transmitter sends a message which can potentially reach all of its neighbors. A node acting as a receiver in a given step gets a message, if and only if, exactly one of its neighbors transmits in this step. If at least two neighbors  $v$  and  $v'$  of  $u$  transmit simultaneously in a given step, none of the messages is received by  $u$  in this step. In this case we say that a *collision* occurred at  $u$ . It is assumed that the effect at node  $u$  of more than one of its neighbors transmitting is the same as that of no neighbor transmitting, i.e., a node cannot distinguish a collision from silence.

In this paper we consider deterministic communication algorithms that use the entire knowledge about the network topology. Such algorithms are useful in radio networks that have a reasonably stable graph of connections. As long as no changes occur in the network topology during the execution of the algorithm, the communication task will be completed successfully. Another interesting aspect of deterministic communication in known radio networks is its close relation with randomized communication in ad-hoc radio networks.

Although either broadcasting or gossiping could be used to solve M2M multicasting, the former often does not scale well while the latter may not be efficient because an application may involve only a small fraction of the total number of nodes of the underlying radio network. In this paper we address the problem of minimizing the communication time of M2M multicast in multi-hop radio networks. To the best of our knowledge, this is the first study of M2M multicast time in this communication model.

## 1.1 Previous Work

Most of the work devoted to radio networks is focused on the broadcasting problem. In the model with known radio network topology, Gaber and Mansour [13] showed that the broadcasting task can be completed in time  $O(D + \log^5 n)$ , where  $D$  is the diameter of the network. Two alternative broadcasting algorithms (superior for small diameters) can be found in [5,20]. The computation of an optimal radio broadcast schedule for an arbitrary network is known to be NP-hard, even if the underlying graph of connections is embedded into a plane [4,22].

Many authors [3,6,7,9,10,12,18,11] studied deterministic distributed broadcasting in ad-hoc radio networks, in which every node knows only its own label, using the model of directed graphs. Increasingly faster broadcasting algorithms working on arbitrary  $n$ -node (directed) radio networks were constructed, the currently fastest being the  $O(n \log^2 D)$ -time algorithm from [11]. (Here  $D$  is the radius of the network, i.e, the longest distance from the source to any other node). On the other hand, in [10] a lower bound  $\Omega(n \log D)$  on broadcasting time was proved for directed  $n$ -node networks of radius  $D$ .

The gossiping problem was not studied in the context of radio networks of known topology, until very recent work of Gąsieniec and Potapov [15]. They study the gos-

siping problem in known radio networks, where each node transmission is limited to unit messages. In this model several optimal and almost optimal  $O(n)$ -time gossiping algorithms are proposed in various standard network topologies, including lines, rings, stars and trees. It is also proved that there exists a radio network topology in which gossiping (with unit messages) requires  $\Omega(n \log n)$  time. Very recently, Gąsieniec *et al.* [16] studied gossiping in known radio networks with arbitrarily large messages, and several optimal gossiping algorithms were proposed for a wide range of radio topologies.

So far, the gossiping problem was mostly studied in the context of ad-hoc radio networks, where the topology of connections is unknown to nodes. In this model, Chrobak *et al.* [9] proposed a fully distributed deterministic algorithm that completes the gossiping task in time  $O(n^{3/2} \log^3 n)$ . For small values of the diameter  $D$ , the gossiping time was later improved by Gąsieniec and Lingas [14] to  $O(nD^{1/2} \log^3 n)$ . Another interesting  $O(n^{3/2})$ -time algorithm, a tuned version of the gossiping algorithm from [9], can be found in [24]. A very recent  $O(n^{4/3} \log n)$ -time gossiping algorithm has been proposed by Gąsieniec *et al.* in [17]. A study of deterministic gossiping in ad-hoc radio networks, with messages of limited size, can be found in [8]. The gossiping problem in *ad-hoc* radio networks also attracted studies based on efficient randomized algorithms. In [9], Chrobak *et al.* proposed an  $O(n \log^4 n)$ -time gossiping procedure. This time was later reduced to  $O(n \log^3 n)$  [21], and very recently to  $O(n \log^2 n)$  [11].

## 1.2 Our Results

The aim of this paper is the design of efficient algorithms for the M2M multicasting problem in radio networks. We study the complexity of this problem for  $k$  participating nodes in an  $n$ -node radio network. While the topology of the network is known to all nodes, participating nodes are not aware of each other's positions. We show that if the maximum distance between any two out of  $k$  participants is  $d$  then this information exchange problem can be solved in time  $O(d \log^2 n + k \log^3 n)$  by a fully distributed deterministic algorithm. Hence our algorithm is linear in the size of the subnetwork induced by the participating nodes, and only polylogarithmic in the size of the entire radio network. Our solution is based on a novel application of the graph clustering method preserving locality [13] and on efficient adaptive collision resolution based on the concept of promoters, see section 2.1.

## 2 Paradigms and Tools

All multicast algorithms presented in this paper are based on the following idea. The nodes participating in the multicast process communicate with other participants via messages. Each participating node has initially one message which is the label of the node. The aim is that all participants learn labels of all other participants.

In the first part of the algorithm, the messages are gathered in one selected *meeting point*. The messages traveling towards the meeting point, from time to time compete with other messages for the same communication channel. We will guarantee the invariant that each message competes with any other message at most once. Moreover, the time spent during any particular competition with  $l$  other messages is bounded by  $O(l \log^2 n)$ .

Note that, although each traversing message is kept in a single copy, it leaves its trace in each visited node. In the second part of the multicast procedure, a compound message containing all individual messages is distributed to all participating nodes to inform them about the labels of the others.

Although the algorithms used for trees and for arbitrary graphs share the same general structure, they differ dramatically in details of their design. The two main differences lie in the choice of the meeting point and in the way in which the competition for the same communication channel is resolved.

In trees, the selection of the meeting point is implicit. Before the communication process is started, one node is chosen as the root of the tree. During the multicast process, all messages generated by the participating nodes traverse towards this root. The meeting point corresponds to the first node which is visited by all messages. In fact, the meeting point is the lowest common ancestor (LCA) of all participating nodes, with respect to the chosen root of the tree. Note that the distance between the LCA and all participating nodes is always limited to  $d$ . Each competition is resolved with the help of a system of synchronized descending selectors.

In arbitrary graphs, the choice (computation) of the meeting point is much more complex. Not knowing the position of participating nodes, we cannot fix the meeting point in advance, since – in the worst case – messages would have to travel along the diameter of the entire network before meeting each other. Instead, we propose a new clustering concept, that allows us to group all participating nodes in one of the clusters with a relatively small diameter, comparable with  $d$ . Each cluster has its own meeting point and a BFS spanning tree rooted in it. In each cluster, similarly as in the case of trees, we try to move all messages from the participating nodes towards the meeting point. However, efficient traversal limited to branches of the BFS tree is not always possible. This is due to the fact that in the cluster there exist edges outside of the BFS tree that potentially cause a lot of conflicts. Thus the competition is becoming much harder. In order to overcome this problem, we propose a special algorithm that resolves conflicts between competing messages. This algorithm is based on a novel use of descending selectors, combined with broadcasting and gossiping procedures.

## 2.1 Resolving Competition

The main difficulty occurring in radio communication is the presence of collisions. It has been shown before, see, e.g., [10,9], that the most efficient tools designed for collision resolution are based on combinatorial structures possessing a *selectivity property*. We say that a set  $R$  hits a set  $Z$  on element  $z$ , if  $R \cap Z = \{z\}$ , and a family of sets  $\mathcal{F}$  hits a set  $Z$  on element  $z$ , if  $R \cap Z = \{z\}$  for at least one  $R \in \mathcal{F}$ . In [10] we can find a definition of a family of subsets of set  $\{0, 1, \dots, N - 1\} \equiv [N]$  which hits each subset of  $[N]$  of size at most  $k \leq N$  on all of its elements. They refer to this family as *strongly  $k$ -selective family*. They also prove the existence of such a family of size  $O(k^2 \log N) = O(k^2 \log n)$ . In [9] we find a definition of a family of subsets of set  $\{0, 1, \dots, N - 1\} \equiv [N]$  which hits each subset of  $[N]$  of size at most  $k$  on at least  $k/2$  distinct elements, where  $N \geq k \geq 1$ . They call it a  *$k$ -selector* and prove the existence of such a family of size  $O(k \log N) = O(k \log n)$ .

In what follows we show how to cope with collisions occurring during the competition process with a help of selective families and selectors.

**Promoting messages in unknown stars.** Assume  $k$  nodes from  $V' = \{v_1, v_2, \dots, v_k\}$  are immediate neighbors (not aware of each other) of another node  $w$ , i.e., they form a star with a center in  $w$ , and they all compete (at some stage of the algorithm) to move their message to  $w$ . The process of moving messages from nodes in  $V'$  to  $w$  is called a *promotion*. It is known, that the mechanism based on the selector idea allows a fraction (e.g., a half) of the nodes in  $V'$  to deliver their messages to  $w$  in time  $O(k \log n)$  [9]. Let  $S(k)$  represent the collision resolution mechanism based on selectors. Note that  $S(k)$ , if applied in undirected networks, can be supported by the *acknowledgment of delivery* mechanism in which each transmission from the neighbors of  $w$  is alternated with an acknowledgement message coming from the central node  $w$ . If during the execution of  $S(k)$  a transmission towards  $w$  is successful, i.e., one of  $v_i \in V'$  succeeds in delivering its message, the acknowledgement issued by  $w$  and returned to all nodes in  $V'$  contains the label of the successful node; otherwise the acknowledgement is null. Let  $\mathbf{S}(k)$  be the mechanism with the acknowledgement feature based on  $S(k)$ . In other words, the use of  $\mathbf{S}(k)$  allows to exclude from further transmissions all nodes in  $V'$  that have managed to deliver their message to  $w$  during the execution of  $\mathbf{S}(k)$ . Note that the duration of  $\mathbf{S}(k)$  is  $O(k \log n)$ , see [9].

Let  $S^*(i)$  be the communication mechanism based on concatenation (superposition) of  $i$  selectors  $S(2^i), S(2^{i-1}), \dots, S(2^1)$ . We will call it later as a *descending selector*. The descending selector extended by the acknowledgement mechanism, i.e., the concatenation of  $\mathbf{S}(2^i), \mathbf{S}(2^{i-1}), \dots, \mathbf{S}(2^1)$ , forms a *promoter* and it is denoted by  $\mathbf{S}^*(i)$ . Note that the duration of  $\mathbf{S}^*(i)$  is  $O(2^i \log n)$ .

**Lemma 1.** *If  $V' = \{v_1, v_2, \dots, v_k\}$  is a set of neighbors of  $w$ , and all nodes in  $V'$  use the same promoter  $\mathbf{S}^*(i)$ , where  $k \leq 2^i$ , then all nodes in  $V'$  deliver their messages to  $w$  in time  $O(2^i \log n)$ .*

*Proof.* The proof is done by induction, and is based on the fact that after the execution of each  $\mathbf{S}(2^j)$ , for  $j = i, \dots, 1$ , the number of competing nodes in  $V'$  is  $\leq 2^{j-1}$ .

**Promoting messages in unknown bipartite graphs.** Assume that we have a connected bipartite graph  $B$  in which nodes are partitioned into two sets  $U$  and  $L$ . In our further considerations, sets  $U$  and  $L$  will correspond to two adjacent BFS levels, upper and lower respectively, in a subgraph of  $G$ . While, in general, nodes in  $U$  and  $L$  are not aware of the presence of each other, we assume here that each node  $x \in L$  is associated with exactly one of its neighbors (called later a *parent*)  $y \in U$  and this relation is known to both of them. Note that a node in  $U$  can be a parent of several nodes in  $L$ , thus  $|U| \leq |L| = l$ . We assume also, that initially only nodes in  $L$  are aware of their presence in  $B$ , i.e., their parents must be informed about it by the children. In what follows we show how to move all messages available at nodes of  $L$ , to a single node in  $U$  in time  $O(l \log^2 n)$ . We first assume that the size  $l$  is known in advance. As in the case of stars, we call the process of moving messages from  $L$  to  $U$  a promotion. The promoting algorithm works in 5 stages.

**procedure** ENHANCED-PROMOTION( $L$ );

1. All nodes in  $L$  contact their parents;  
(level  $U$  is formed).
2. All nodes belonging to  $B$  take part in leader election choosing a node  $r$  among all nodes in  $U$ ;  
(node  $r$  is going to collect all messages initially stored in  $L$ ).
3. Node  $r$  initiates broadcasting to all other nodes in  $B$ ;  
(the broadcast tree (with unidirectional edges) rooted in  $r$  is created).
4. Each node (except the root  $r$ ) contacts its parent in the broadcasting tree;  
(bidirectional edges are now available in the broadcast tree).
5. The root  $r$  sends a token visiting all nodes of the broadcasting tree to collect all messages from  $L$  and place them in  $r$ ;  
(all messages are gathered in  $r$ ).
6. The root  $r$  sends a token visiting all nodes of the broadcasting tree, in order to confirm successful delivery of every competing message.

Step 1 is based on a single use of the promoter  $S^*(i)$ , for  $i - 1 < \log l \leq i$ . Even if promoters are designed primarily for promoting nodes in stars, they also prove to be useful in the case of bipartite graphs (with established parent/child relation). As before, we say that a node  $x \in L$  contacts its parent  $y$  successfully, when all other nodes in  $L$  remain silent. This means that the acknowledgement which is later sent by  $y$ , will not collide with other messages. The time of step 1 is  $O(l \log n)$ .

Step 2 is based on the leader election algorithm from [9] combined with the very recent fast deterministic broadcasting algorithm in [19]. The election algorithm works in time  $O(l \log^2 n)$ .

Step 3 is based on the broadcasting algorithm presented in [6] and works in time  $O(l \log n)$ .

Step 4 is analogous to Step 1. This gives the time complexity  $O(l \log n)$ .

Steps 5 and 6 are implemented as a simple tree (e.g., pre-order) traversal in time  $O(l)$ , for details see [6].

Thus the total time of the algorithm is bounded by  $O(l \log^2 n)$ .

## 2.2 Graph Clustering Preserving Locality

The main purpose of the clustering method is to obtain a representation of a large graph as a collection of its much smaller subgraphs (clusters), while preserving local distances between the nodes.

Let  $G = (V, E)$  be a graph representing a radio network. Initially we pick an arbitrary node  $c$  in  $V$  that becomes a *central node* in  $G$ . The radius of  $G$  is the maximum distance  $D$  between  $c$  and any other node. The clustering method groups nodes belonging to some connected subgraphs  $G'$ , in the same cluster  $C$ . If the diameter of  $G'$  is  $d$ , the diameter of  $C$  is at most  $O(d \log n)$ .

**Definition 1.** Let  $l_j$  be the  $j^{\text{th}}$  BFS level in a graph  $G$  with respect to a central node  $c$ , i.e.,  $l_j = \{v \mid \text{dist}(c, v) = j\}$ .

**Definition 2.** A partition  $\pi(x)$  of the graph  $G$  is a division of  $G$  into super-levels, such that, each super-level is composed of  $4d$  consecutive BFS levels, where the first super-level starts from an arbitrary but fixed BFS level  $l_x$  (note that levels  $l_0, l_1, \dots, l_{x-1}$  are excluded from the partition  $\pi(x)$ ). More formally, the  $i$ th super-level in  $\pi(x)$  is  $G_i(x) = \{v | v \in l_j, (i - 1 - x) \cdot 4d \leq j \leq (i - x) \cdot 4d - 1\}$ , for  $i = 1, 2, \dots, \lceil \frac{D-x}{4d} \rceil$ , where  $D$  is the radius of  $G$  with respect to the central node  $c$ . Given a super-level  $G_i(x)$ , its top level is  $l_{(i-1-x) \cdot 4d}$ , and its bottom level is  $l_{(i-x) \cdot 4d - 1}$ . Note that  $G_i(x)$  is not necessarily connected.

**Definition 3.** For each node  $u$  belonging to the top level of  $G_i(x)$ , we define the pre-cluster  $S_u^{(i)}$ , which contains all nodes in  $G_i(x)$  at distance  $\leq 4d$  from  $u$ .

**Definition 4.** The clusters are obtained by growing appropriate pre-clusters, according to the mechanism used in the Cover Algorithm presented in [13]. In short, the growing algorithm is performed in  $O(\log n)$  stages. In each stage  $i = 1, \dots, \log k$  a collection of clusters  $C_*^i$  (each at distance 2 apart) is created as follows. We start with an arbitrary (yet available) pre-cluster which forms a core of a new cluster  $C_0^i$ . At each step of the extension procedure we add to the cluster  $C_0^i$  a new layer of pre-clusters that intersect with  $C_0^i$  or are at distance at most 1 from  $C_0^i$ . Note that this extension is successful only if the number of new nodes coming with the new pre-clusters is at least as big as the number of nodes in the pre-clusters already present in the cluster  $C_0^i$ . If this condition is not met, the extension of the cluster  $C_0^i$  is terminated, i.e., the construction of  $C_0^i$  completes without augmenting nodes available in the just considered layer of pre-clusters. Instead, the pre-clusters in the new layer are moved for consideration in stage  $i + 1$ . The process of growing clusters  $C_1^i, C_2^i, \dots$  is performed similarly, and it continues as long as we have at least one pre-cluster that neither forms a part of any cluster constructed in stages  $1, \dots, i$ , nor has been moved for consideration in stage  $i + 1$ .

**Lemma 2.** The clusters have the following properties:

1. Each cluster is a union of some pre-clusters,
2. Each pre-cluster is a member of exactly one cluster.
3. Each cluster is a connected sub-graph of  $G$ .
4. The diameter of each cluster is  $O(d \log n)$ , and
5. There is a  $O(\log n)$ -colouring of the clusters, such that, clusters having the same color are at distance  $\geq 2$  apart.

*Proof.* Properties 1, 2, and 3 follow directly from the construction of the clusters. Property 4 is based on the fact that each pre-cluster has diameter  $\leq 4d$  and that during construction of any cluster the number of new layers of pre-clusters is limited to  $\log n$ , since each extension by a new layer of pre-clusters at least doubles the number of nodes in the pre-clusters of currently constructed cluster. Property 5 follows from the fact that during each round we construct clusters at distance 2 apart. Note also that the number of rounds is bounded by  $\log n$ . This is because in each round at least half of the nodes available in pre-clusters is used to build the clusters of the same color. This is a consequence of arguments used in the proof of Property 4.

**Definition 5.** *The 2-partition of the graph  $G$  comprises two different partitions:  $\pi(0)$  which starts at the super-level  $G_1(0)$ , and  $\pi(2d)$  which starts at the super-level  $G_1(2d)$ .*

**Lemma 3.** *In at least one of the partitions of the 2-partition, there exists at least one cluster that contains all  $k$  participating nodes and the shortest paths between them. Moreover, in this partition, any other cluster containing some (or all) of the  $k$  points, is colored differently.*

*Proof.* Let  $v$  be one of the  $k$  points. According to our definition of the 2-partition, we can prove that the node  $v$  must fall into the central  $2d$  BFS levels of a super-level in one of the partitions, except for the case when  $v$  belongs to the first  $d$  BFS levels (when all  $k$  points belong to the cluster based on the central node  $c$ ). Thus, there exists a node  $p$  at the top level of the corresponding super-level  $G_i(\cdot)$ , which is at distance  $\text{dist}(p, v) \leq 3d$  from the node  $v$ . Since all other participating nodes are at distance  $\leq d$  from  $v$ , there exists a pre-cluster (which constitutes a part of a cluster)  $S_p^{(i)}$  which contains the entire set of  $k$  participating nodes. The second part of the lemma follows from the fact that clusters having the same color cannot overlap.

### 3 Efficient M2M Multicast

We start this section with the presentation of a M2M multicasting procedure designed for radio networks with the tree topology. M2M multicast in trees works in time  $d + O(k \log^2 n)$ -time. We later present a more complex M2M multicast procedure which works in an arbitrary topology in time  $O(d \log^2 n + k \log^3 n)$ .

#### 3.1 M2M Multicast in Trees

Our M2M multicast algorithm is based on the following principle. The participating nodes make aware other nodes (including all other participants) about their presence by distributing appropriately aimed messages. These are initially gathered in a selected, *central node*, and then distributed to all other participating nodes.

The outline of the multicast algorithm is presented below.

**procedure** TREE-MULTICAST( $T$ )

1. All nodes agree on the root  $r$  of the tree  $T$ ;  
(the nodes of the tree  $T$  are now divided into BFS levels with respect to the distance from the root  $r$ ).
2. Messages issued by the participating nodes traverse, level by level, towards  $r$ ;  
(traces left by the messages at the intermediate nodes meet eventually, at the latest in  $r$ ).
3. The first node that is visited by all  $k$  messages, called the *meeting point*, distributes the compound message back towards all participating nodes;  
(this completes the multicast process).

Step 1., is straightforward. Since all nodes know the topology of  $G$  (including the labels of nodes), they use the same deterministic algorithm to choose the root  $r$  (e.g., the node with the smallest label). There is no communication involved in this step.

Step 2. is based on synchronized use of promoters and certain properties of rooted trees. Note that during the traversal, a message may meet other messages and compete, e.g., for the access to the same parent in the BFS tree. There may also be collisions caused by simultaneous transmissions at adjacent BFS levels. The latter problem can be solved by enforcing an extra rule that nodes at BFS level  $j$  (at distance  $j$  from the root  $r$ ) execute their transmissions in steps  $i$ , where  $i = j \pmod 3$ . This slows down the whole process only by a multiplicative constant 3. The problems caused by the competition of messages require more careful consideration. When the control messages traverse towards the root  $r$  of the tree  $T$ , each successful transmission must be always confirmed (see the definition of promoters in section 2.1). If the acknowledgement arrives, the transmission is considered to be successful. Otherwise, a special promotion mechanism is switched on, which is designed to deal with the message competition. In what follows we assume that a message uses different (interleaved) time slots for *fast transmissions* (associated with an immediate acknowledgement) and slow transmissions (associated with the competition).

In the promotion mechanism, we use exactly  $\log k$  promoters  $\mathbf{S}^*(1), \dots, \mathbf{S}^*(\log k)$  that are run “simultaneously” and periodically. The “simultaneous” execution of promoters of different sizes is done by the time multiplexing, i.e., the execution of two consecutive transmission steps in any  $\mathbf{S}^*(i)$  is interleaved with the execution of single steps of every other promoter. Moreover the execution of the promoters of different sizes is synchronized, i.e., a single execution of the promoter  $\mathbf{S}^*(i)$  corresponds to two executions of the promoters  $\mathbf{S}^*(i-1)$ , for any  $i = 2, \dots, \log k$ . Any message traversing towards the root  $r$ , when it enters the promotion mechanism at some BFS level, it starts using promoter  $\mathbf{S}^*(2)$  as soon as it is available, i.e., when the new execution of  $\mathbf{S}^*(2)$  is scheduled. At the end of the execution of  $\mathbf{S}^*(2)$ , if the message is not promoted to the next level, it starts using promoter  $\mathbf{S}^*(4)$  as soon as it is available. This means that it may wait  $|\mathbf{S}^*(2)|$  time steps before the new execution of  $\mathbf{S}^*(4)$  takes place. In general, the message can wait for the execution of  $\mathbf{S}^*(i)$  at most  $|\mathbf{S}^*(i-1)|$  time steps. Note that, when the number of competing messages is bounded by  $2^i$ , all messages are promoted after the execution of  $\mathbf{S}^*(2^i)$ . Since the running time of all previously used (smaller) promoters and the waiting time is bounded by  $(2|\mathbf{S}^*(1)| + \dots + 2|\mathbf{S}^*(i-1)|) \cdot O(\log n)$  (including time multiplexing), the total time used to promote the competing messages is  $O(2^i \log^2 n)$ .

**Lemma 4.** *The last message enters the meeting point (the lowest common ancestor (LCA) of all participating nodes, with respect to  $r$ ) in time  $O(d + k \log^2 n)$ .*

*Proof.* Note that the *lowest common ancestor* (LCA) of all participating nodes (with respect to  $r$ ) is at distance at most  $d$  from each of them. Consider a single message. When it moves towards the root (in fact, towards the meeting point LCA), it traverses each edge in two time units, if there is no competition. The time complexity related to this type of transmissions can be bounded by  $O(d)$ . If at any time the message competes with some other  $l$  messages, it is promoted to the next BFS level in time  $O(l \log^2 n)$ . Note that two messages competing once will never compete against each other again,

since later on, they travel along the same path towards the root of the tree. This means that the total time spent by a message on competing with other messages is bounded by  $O(k \log^2 n)$ . Thus the last message arrives at the meeting point in time  $O(d + k \log^2 n)$ .

Step 3. is a simple broadcasting procedure that distributes the compound message to all nodes (including all participants) within distance  $d$  from the meeting point. Since there are no collisions in radio broadcasting in trees, the compound message is distributed to all participating nodes in time at most  $d$ .

**Theorem 1.** *The M2M multicast problem in radio networks with a tree topology can be solved in time  $O(d + k \log^2 n)$ .*

### 3.2 M2M Multicast in Arbitrary Graphs

In this section we show how to perform M2M multicast in arbitrary radio networks in time  $O(d \log^2 n + k \log^3 n)$ . The algorithm is based on the clustering method introduced in section 2.2, on efficient promotion of messages in bipartite graphs, see section 2.1, and some other observations.

In view of the clustering method, there exists at least one (and at most  $\log n$ ) cluster(s) with diameter  $\leq d \log n$  that contain(s) all  $k$  participating nodes. In what follows, we consider computation performed inside a single cluster. Recall that simultaneous execution of transmissions in clusters having the same color does not cause collisions between the clusters, because all clusters of the same color are at distance at least 2 apart. In order to avoid collisions between clusters in different colors, we execute computation for different colors in  $O(\log n)$  (number of colors) different stages. This gives an  $O(\log n)$  slowdown in comparison with an execution in a single cluster. Note that having the partition into clusters ready, we could now perform the M2M multicast in time  $O(k \cdot d) \text{polylog} n$ , applying a leader election algorithm and broadcasting  $k$  times. However, our intention is to design a  $O((k + d) \text{polylog} n)$  algorithm (thus linear in the size of the subnetwork induced by the participating nodes and only polylogarithmic in the size of the entire radio network). The computation in a cluster  $C$  of the 2-partition is performed as follows.

**procedure** GRAPH-MULTICAST( $C$ )

1. Select a leader in  $C$  which becomes the root  $r$  of a spanning BFS tree  $T$ ;  
(after this step the nodes in  $C$  are partitioned into BFS levels with respect to the distance from the root  $r$ ).
2. Messages sent by the participating nodes travel, level by level, towards the root  $r$ ;  
(note that, in the case of a competition, a message may be routed to the next BFS level via (a sequence) of edges, including those not belonging to the BFS tree  $T$ ).
3. The root  $r$  distributes the compound message to all participating nodes;  
(This completes the multicast process).

Step 1. does not involve communication, since the topology of  $G$  is known to every node. Thus the division of  $G$  into clusters can be computed locally and independently in each node of  $G$ .

Step 2. uses two types of moves. Some moves towards the root are performed along the edges of the BFS tree. However, such simple moves are feasible only in the case when the traversing messages are not involved in any competition. As soon as a traversing message starts to compete (i.e., it does not receive the acknowledgement of the successful transmission), it enters the system of promotion procedures, which is based on the concept of the ENHANCED-PROMOTION procedure, see section 2.1.

The promotion algorithm in arbitrary graphs is more complex than its tree counterpart, due to the presence of external edges (with respect to the BFS tree) that cause more collisions during transmissions. This time, the competition does not always concern a single node that is a joint parent of nodes containing the competing messages. In fact, some nodes containing traversing messages and their parents may form a connected bipartite subgraph  $B$  of  $G$  (with partitions  $U$  and  $L$  at adjacent BFS levels). Regardless of the latter difference, we would like to use a similar amortization argument, while assessing the time complexity of the multicast algorithm. Indeed, we show that if at any BFS level,  $l$  messages are involved in the competition (within a bipartite graph  $B$ ), all messages from the set  $L$  will be moved to a single node in  $U$  in time  $O(l \log^2 n)$ . Thus if two messages compete once in some bipartite graph, they will never compete against each other again. Similarly as in the case of trees, the promoting algorithm is based on simultaneous (interleaved) and periodic execution of the procedure ENHANCED-PROMOTION( $i$ ), for  $i = 1, \dots, \log k$ , that deals with sets of competing messages of size  $2^1, 2^2, \dots, 2^{\log k}$ , respectively. Recall that in section 2.1 we explained how to promote competing messages in bipartite graphs, when the size of the set of competing messages is known. In what follows we explain how this assumption can be dropped and shed more light on details of the promotion algorithm used at any BFS level.

At any BFS level, when a message  $m$  traversing towards the root  $r$  enters the promotion mechanism, it waits for the first available execution of the procedure ENHANCED-PROMOTION(1). Similarly as in trees, if the promotion was not successful (the number of competitors was too large), message  $m$  waits for the next (complete) execution of the procedure ENHANCED-PROMOTION(2), and so on, for all consecutive powers of two  $\leq \log k$ . Note that in trees, since all messages compete for the same parent, any message promoted to the next level, will never be obstructed by its former competitors again. We would like to use the same invariant in the case of general graphs too. Thus we insist that all messages competing in a bipartite graph eventually meet in one of the nodes of the set  $U$ . Moreover, we will exclude from promotion all messages that managed to gather in one node of  $U$ , if not all their competitors in the bipartite graph  $B$  managed to do so. This is to guarantee that a pair of messages that competed once will never compete again.

Recall that, upon the completion of procedure ENHANCED-PROMOTION( $i$ ), the acknowledgement confirming a successful promotion of all competing messages is sent across the connected component of the bipartite graph  $B$ . If the acknowledgement does not arrive (e.g., when the graph  $B$  is larger than  $2^i$ ), all nodes in  $B$  know that they have to use the next available execution of the procedure ENHANCED-PROMOTION( $i + 1$ ). However, if the confirmation arrives, the competing messages are still not sure whether all messages in  $B$  were properly discovered.

Indeed, there might be several connected components  $B_1, B_2, \dots, B_m$  of  $B$ , satisfying  $B_1 \cup B_2 \cup \dots \cup B_m = B$ , that are not aware of each other at the end of the execution

of ENHANCED-PROMOTION( $i$ ). This happens when, for some reason, all internal transmissions in each  $B_i$  are not interrupted by local transmissions in other components. This can be checked in the following way. Every component  $B_i$  has its leader  $l_i$  whose label will play the role of a label of the whole component  $B_i$ . The pattern of transmissions used in each  $B_i$  is based on the combination of the concept of strongly 2-selective family [10] and of Steps 5 and 6 in the ENHANCED-PROMOTION procedure. One set  $R$  in the strongly 2-selective family, in relation to the label  $l_i$ , is replaced by either the whole execution of Steps 5 and 6 in the ENHANCED PROMOTION procedure (if  $l_i \in R$ ) or by a continuous sequence of *noisy calls* (if  $l_i \notin R$ ), meant to blur communication in the neighboring component. Note that if the component  $B_i$  is connected by an edge with some other component  $B_j$ , there will be a step in the application of the strongly 2-selective family when the bit associated with  $B_i$  is set to 1 and the bit associated with  $B_j$  is set to 0 (and vice versa). In this case the traversal of the message in the component  $B_i$  will be interrupted, which is enough to figure out that  $B_i$  does not form the whole graph of competitors. The cost of Steps 4 & 5 is bounded by  $O(2^i)$  and the number of steps in the strongly 2-selective family is  $O(\log n)$ . Thus the cost of this test (including time multiplexing) is bounded by  $O(2^i \log^2 n)$ .

In Step 3, the distribution of the compound message is performed with the help of a broadcasting procedure from [20] in time  $O(d \log n + \log^2 n)$ .

**Theorem 2.** *The M2M multicast problem in arbitrary radio networks can be solved in time  $O(d \log^2 n + k \log^3 n)$ .*

## 4 Conclusion

In this paper we gave an  $O(d \log^2 n + k \log^3 n)$ -time algorithm for solving the M2M multicast problem for a group of  $k$  participating nodes with maximum distance  $d$  in an arbitrary radio network consisting of  $n$  nodes. Our approach uses a clustering technique for partitioning the radio network and a new algorithm for promoting messages in clusters. Interesting problems left for further investigation include (1) improving the upper bounds of our algorithms, (2) developing locality-sensitive multicast algorithms for the case when the nodes of the network have only limited (e.g., local) knowledge of the topology, and (3) investigating how efficient updating affects performance of multicast in mobile radio systems.

## References

1. S. Banerjee, S. Khuller, A Clustering Scheme for Hierarchical Control in Multi-hop Wireless Networks, in Proc. *INFOCOM 2001*, pp 1028-1037.
2. R. Bar-Yehuda, O. Goldreich, and A. Itai, On the time complexity of broadcast in multi-hop radio networks: An exponential gap between determinism and randomization, *Journal of Computer and System Sciences*, 45 (1992), pp 104-126.
3. D. Bruschi and M. Del Pinto, Lower bounds for the broadcast problem in mobile radio networks, *Distributed Computing* 10 (1997), pp 129-135.
4. I. Chlamtac and S. Kutten, *On broadcasting in radio networks-problem analysis and protocol design*, *IEEE Transactions on Communications* 33 (1985), pp 1240-1246.

5. I. Chlamtac and O. Weinstein, The wave expansion approach to broadcasting in multihop radio networks, *IEEE Trans. on Communications* 39 (1991), pp 426-433.
6. B. Chlebus, L. Gąsieniec, A. Gibbons, A. Pelc and W. Rytter, Deterministic broadcasting in unknown radio networks, *Distributed Computing* 15 (2002), pp 27-38.
7. B. Chlebus, L. Gąsieniec, A. Ostlin, and M. Robson, Deterministic Radio Broadcasting, in Proc. *27th Int. Colloq. on Automata, Languages and Programming*, ICALP'00, pp 717-728.
8. M. Christersson, L. Gąsieniec and A. Lingas, Gossiping with bounded size messages in ad-hoc radio networks, in Proc. *29th International Colloquium on Automata, Languages and Programming*, ICALP'02, pp 377-389.
9. M. Chrobak, L. Gąsieniec and W. Rytter, Fast Broadcasting and Gossiping in Radio Networks, *Journal of Algorithms* 43(2), 2002, pp 177-189.
10. A.E.F. Clementi, A. Monti and R. Silvestri, Selective families, superimposed codes, and broadcasting on unknown radio networks, in Proc. *12th Ann. ACM-SIAM Symposium on Discrete Algorithms*, SODA'01, pp 709-718.
11. A. Czumaj and W. Rytter, Broadcasting algorithms in radio networks with unknown topology, in Proc. *44th Ann. Symp. on Foundations of Computer Science*, FOCS'03, pp 492-501.
12. G. DeMarco and A. Pelc, Faster broadcasting in unknown radio networks, *Information Processing Letters* 79, 2001, pp 53-56.
13. I. Gaber and Y. Mansour, *Broadcast in radio networks*, in Proc. *6th Ann. ACM-SIAM Symp. on Discrete Alg.*, SODA'95, pp 577-585. Also, *Journal of Algorithms*, 46(1), 2003, pp 1-20.
14. L. Gąsieniec and A. Lingas, On adaptive deterministic gossiping in ad hoc radio networks, *Information Processing Letters* 2(83), 2002, pp 89-94.
15. L. Gąsieniec and I. Potapov, *Gossiping with unit messages in known radio networks*, in Proc. *2nd IFIP Int. Conference on Theoretical Computer Science*, TCS'02, pp 193-205.
16. L. Gąsieniec, I. Potapov and Q. Xin, Time efficient gossiping in known radio networks, to appear in Proc. *11th Colloq. on Struct. Inform. and Comm. Complexity*, SIROCCO'04.
17. L. Gąsieniec, T. Radzik and Q. Xin, Faster deterministic gossiping in *ad-hoc* radio networks, to appear in Proc. *9th Scandinavian Workshop on Algorithm Theory*, SWAT'04.
18. D. Kowalski and A. Pelc, Faster deterministic broadcasting in ad hoc radio networks, in Proc. *20th Ann. Symp. on Theor. Aspects of Comp. Science*, STACS'03, pp 109-120.
19. D. Kowalski and A. Pelc, Broadcasting in undirected ad hoc radio networks, in Proc. *22nd ACM Symposium on Principles of Distributed Computing*, PODC'03, pp 73-82.
20. D. Kowalski and A. Pelc, Centralized deterministic broadcasting in undirected multi-hop radio networks, manuscript 2004.
21. D. Liu and M. Prabhakaran, On Randomized Broadcasting and Gossiping in Radio Networks, in Proc. *8th Annual International Conference on Computing and Combinatorics*, COCOON'02, pp 340-349.
22. A. Sen and M.L. Huson, *A new model for scheduling packet radio networks*, in Proc. *15th Ann., Joint Conference of the IEEE Comp. and Comm. Soc.*, 1996, pp 1116-1124.
23. P.J. Slater, E.J. Cockayne and S.T. Hedetniemi, Information Dissemination in Trees, *SIAM Journal on Computing*, 10, 1981, pp 892-701.
24. Y. Xu, An  $O(n^{1.5})$  deterministic gossiping algorithm for radio networks, *Algorithmica*, 36(1), 2003, pp 93-96.