An Optimal Randomized Broadcasting Algorithm in Radio Networks with Collision Detection

Ny Aina ANDRIAMBOLAMALALA\textsuperscript{1} and Vlady RAVELOMANANA\textsuperscript{2}

\textsuperscript{1}MISA — University of Antananarivo — Madagascar.
\textsuperscript{2}IRIF UMR CNRS 8243 — University Denis Diderot — France.

Abstract

We present a randomized distributed algorithm that in radio networks with collision detection broadcasts a single message in $O(D + \log^2 n)$ time slots, with high probability\textsuperscript{1}. In view of the lower-bound $\Omega(D + \log^2 n)$, our algorithm is optimal in the considered model answering the decades-old question of Alon, Bar-Noy, Linial and Peleg in [2].

1 Introduction

In this paper we consider the fundamental problem in distributed computing of broadcasting in radio networks. Such wireless network is usually modeled by an undirected graph and communication operates in synchronous rounds. In the classical broadcasting problem, one distinguished source node $s$ has a message $M$ that needs to be sent to all the other nodes in the network.

During each round or time slot, each node can either listen or broadcast a message of size $O(\log n)$ bits to all of its neighbors. A characteristic of radio networks is that multiple messages that arrive at a node simultaneously collide with one another and none of them is received successfully. Regarding whether nodes can distinguish such a collision from complete silence, the model falls usually into two categories: with collision detection and without collision detection.

For the model with collision detection we consider here, a node receives a message in a given time slot if and only if it keeps silent and exactly one of its neighbors transmits in the same round. If none of its neighbors transmits, a node hears nothing but if more than one neighbor — including itself — transmits a collision occurs and the processor hears noise.

If it has been proved that many fundamental problems (such as leader election [20, 11]) can often be solved faster in the more powerful model of radio networks with collision detection, for the broadcasting problem (see for instance [19]) it had remained unclear whether this is also true until the recent result of Ghaffari, Haeupler and Khabbazian [10] achieving the broadcasting task in $O(D + \log^6 n)$ w.h.p. thus improving the $O(D \log^2 \frac{n}{D} + \log^2 n)$ algorithms due to Czumaj and Rytter [7] and Kowalski and Pelc [14].

\textsuperscript{1}In this paper, an event that occurs “with high probability” (w.h.p. for short) is one whose probability is at least $1 - \frac{1}{n^c}$, for a constant $c \geq 1$, and where $n$ is the network size.
Let us quote an important remark by Alon, Bar-Noy, Linial and Peleg in their seminal work ([2, p. 291]) stating that:

“We cannot rule out the possibility that an \(O(D + \log^2 n)\) schedule always exists. This is a problem on efficient mechanisms for pipelining message passing: Let \(V_i\) be the set vertices at distance \(i\) from the sender \(s\); the network may be engaged in passing \(M\) from \(V_i\) to \(V_{i+1}\), while dealing with \(V_j\) and \(V_{j+1}\), for some \(j > i + 1\). How efficiently this may be done we do not know, and the question is quite intriguing.”

Thus, finding a schedule or equivalently an algorithm that achieves the broadcasting task in time \(O(D + \log^2 n)\) or proving that such an algorithm does not exist appears to be a fundamental research task left open for decades.

Here, we present for the first time an optimal randomized algorithm for the problem of broadcasting in radio networks with collision detection. As in many previous papers, we assume that all nodes have a unique identifier. We also assume that the nodes know a polynomial upper bound on the number of participants \(n\) and a constant factor upper bound on diameter \(D\). Under these assumptions\(^2\), our protocol solves the broadcasting problem in \(O(D + \log^2 n)\) time slots w.h.p.

1.1 Related works

Distributed broadcast algorithms have been studied in depth over the past years. For a comprehensive review about single-message broadcast, we refer to Peleg’s survey [19]. In their pioneering paper, Bar-Yehuda, Goldreich and Itai [3] presented the DECAY protocol, which was extensively used as a fundamental primitive for over the last thirty years when working on multi-hop radio networks. The DECAY protocol broadcasts a message in \(O(D \log n + \log^2 n)\) steps in the model without collision detection. Optimizing DECAY, Kowalski and Pelc [14], and independently Czumaj and Rytter [7] designed \(O(D \log \frac{n}{D} + \log^2 n)\) algorithms. According to the lower-bounds \(\Omega(\log^2 n)\) in [2] and \(\Omega(D \log \frac{n}{D})\) in [16], the algorithms in [7] and [14] are optimal in the model without collision detection. During ten years, no better algorithm was known for the model with collision detection prior to the remarkable result of Ghaffari, Haeupler and Khabbazian [10] working in \(O(D + \log^6 n)\) which is also the first algorithm working in linear time w.r.t. \(D\). Note that Haeupler and Wajc [12] conceived a radio broadcast primitive with complexity \(O(D \log n \log \log n + \log^{O(1)} n)\) w.h.p. Namely, their protocol works also in the model without collision detection, thus improving over the DECAY protocol of [3], and is near-optimal for \(D = n^\epsilon\). Assuming that the topology of the network is totally known, Gasieniec, Peleg and Xin [8] wrote an optimal algorithm running in \(O(D + \log^2 n)\) time slots and Kowalski and Pelc [15] designed a deterministic broadcast protocol with the same complexity.

1.2 Our result

The present work answers the above long-standing research question positively by showing that:

\(^2\)If \(N\) denotes the polynomial known upper-bound on \(n\), we assume that \(\log n = C \log N\) for some constant \(C\). For sake of simplicity, we use abusively the “\(\log n\)” term in all our analyses that should involve the “\(\log N\)” term. A similar remark holds for the diameter \(D\).
Theorem 1 In radio networks with collision detection of size $n$ and diameter $D$, there exists a randomized distributed algorithm that broadcasts a message in $O(D + \log^2 n)$ rounds with high probability.

1.3 Terminology

The radio network is modeled as an undirected graph $G = (V, E)$ with $|V| = n$. The source node is denoted $s$. The message to be transmitted from $s$ to all the nodes of the network is denoted $M$. Each node has and know its unique identifier.

Throughout the rest of this paper, $\lambda$ is a fixed integer greater than or equal to 4 known by all the nodes of the networks (all the participating nodes should agree on the value of $\lambda$). We assume that $D$ divides $\lambda$.

A node is said active when it has received the original message $M$.

The level of a node $v$, denoted $h(v)$, is its hop-distance from the source $s$:

$$h(v) = \text{dist}(v, s).$$

A network is said layered if all its nodes know their respective levels.

Define a layer $\ell$ to be the induced subgraph built with the set of nodes at distance $\ell$ from $s$. We denote such a set $L(\ell)$, that is $L(\ell) = \{v \text{ such that } h(v) = \ell\}$.

The subnetwork $S_i$ ($i \geq 1$) is the induced subgraph of the set of nodes $S_i = \{v \text{ such that } h(v) \in [(i-1)\lambda, i\lambda - 1]\}$. That is $S_i = L((i-1)\lambda) \cup L((i-1)\lambda + 1) \cup \cdots \cup L(i\lambda - 1)$.

Given a subnetwork $S_i$, its frontier consists of the set of nodes at the level $i\lambda - 1$ union the set of nodes at level $(i-1)\lambda + j$ that have no neighbors in the following layer $(i-1)\lambda + j + 1$ (for some $j \in [0, \lambda - 2]$).

A forest of trees of a subnetwork $S_i$ is called skeleton trees if each tree of the forest has its root at the lowest level $(i-1)\lambda$ and its leaves at the frontier of the subnetwork $S_i$.

The figures below depict these definitions with $\lambda = 5$, so that the second subnetwork $S_2$ is the union of $L(5), L(6), L(7), L(8)$ and $L(9)$. The skeleton trees (here a single tree) of $S_2$ are drawn in bold red. The source is somewhere on the left at a distance of 4 hops from the nodes 12, 14 and 23.

**Figure 1.** The ‘empty’ subnetwork $S_2$.

**Figure 2.** The same subnetwork $S_2$ layered and with its skeleton trees.
2 The algorithm

2.1 Overview of the algorithm

To achieve our goal, we devise an algorithm with the following main steps.

a) Layering the network. We start by assigning each node \( v \) a level number \( h(v) \) which represents the hop distance between \( v \) and the initial source \( s \) (this can be done using for instance the beep waves tool described in [9, paragraph 5.3]).

b) Divide into subnetworks. We then divide the network into subnetworks \( \mathcal{S}_1, \cdots, \mathcal{S}_n \) by assigning each node \( v \) with \( h(v) \in [(i-1)\lambda, i\lambda-1] \) to the subnetwork \( \mathcal{S}_i \), each subnetwork consisting of \( \lambda \) consecutive layers. Note that for \( i > \frac{n}{\lambda} \), the subnetworks \( \mathcal{S}_i \) are empty but this does not affect the performance of our algorithms.

c) Parallel computation of skeleton trees. Next, in parallel we build the skeleton trees inside each subnetwork. Observe that this set of trees has the collision-freeness property since any message sent from a node \( v \) property since any message sent from a node \( v \) to its neighbors of the next adjacent level (\( \ell \) adjacent layer). Consider the subgraph whose vertices are these two sets of nodes and \( v \). Thus allowing to build the skeleton trees, by joining iteratively consecutive collision-free sets, layer after layer, inside each subnetwork in \( O(\log^2 n) \) iterations (since \( \lambda \) is a constant).

To construct these skeleton trees, we design a new primitive called Mcfs (standing for Maximal Collision-Free Senders) which plays a central key role in our algorithm. The main goal of Mcfs is to find the maximal collision-free senders \( \{S_1, S_2, \cdots, S_t\} \) in a given layer \( \mathcal{L}(\ell) \) with respect to a given set of receivers \( \{R_1, R_2, \cdots, R_k\} \) in an adjacent layer \( \mathcal{L}(\ell+1) \) or \( \mathcal{L}(\ell-1) \) depending on the needs on the task to be completed. In other words, consider the nodes of the layer \( \mathcal{L}(\ell) \) and the set of receivers \( \{R_1, R_2, \cdots, R_k\} \) (in an adjacent layer). Consider the subgraph whose vertices are these two sets of nodes and an edge \( e \) is drawn if and only if \( e \) connects 2 nodes between 2 different layers. In this subgraph, Mcfs aims to compute a subset \( \{S_1, S_2, \cdots S_t\} \) of the layer \( \ell \) which should be maximal with respect to the collision-freeness property when sending messages to the receivers \( \{R_1, R_2, \cdots, R_k\} \). That is there is no node outside the set \( \{S_1, S_2, \cdots, S_t\} \) that may join it if we want to send messages without interference from the layer of the \( S_t \) to the set of receivers \( R_j \).

We will see that the algorithm Mcfs computes a collision-free set in \( O(\log^2 n) \) w.h.p. thus allowing to build the skeleton trees, by joining iteratively consecutive collision-free sets, layer after layer, inside each subnetwork in \( O(\log^2 n) \) iterations (since \( \lambda \) is a constant).

It is important to observe that the skeleton trees can be computed independently in parallel by the subnetworks.

d) Connecting the skeleton trees in parallel. In the fourth main phase, we use again the Mcfs protocol to build collision-free connections, also in parallel, between:
- the leaves of the skeleton trees in the subnetwork \( \mathcal{S}_i \) and
- the roots of the skeleton trees in \( \mathcal{S}_{i+1} \).

e) Sending from the source to the skeleton trees. After phase d), we are now able to send the original message \( M \) from the source \( s \) to the nodes of the skeleton trees of the first subnetwork \( \mathcal{S}_1 \) in \( \lambda \) time slots using only the nodes of the trees (since there is no interference). The same procedure can be used as the leaves of the trees of \( \mathcal{S}_1 \) have been connected to the roots of the trees of \( \mathcal{S}_2 \) (by the zig-zag gluing procedure which will be detailed in ??) and so on. At the end of this phase, w.h.p any node of any subnetwork is at hop-distance at most \( 2 \times \lambda \) of another active node (node that already contains the message \( M \) from the source \( s \)).
f) Flooding. To broadcast $M$ to the whole network, it suffices that each active node floods the inactive ones by invoking the *Decay* protocol (since it solves the $d$-hop broadcast problem in the radio network model in $O(d \log n + \log^2 n)$ rounds with high probability [3]) repeatedly during a constant number of time slots (say $4 \times \lambda$ rounds).

2.2 Layering and subdividing the network

We use the so called beep waves tool from [9, paragraph 5.3] to take full advantage of the collision detection ability. We want to assign each node $v$ a level number $h(v) = \text{dist}(v, s)$ which represents the (hop) distance between $v$ and $s$. The waves start from the source $s$ at time $t$ by sending a bit and each node $u$ that detects noise (collision) or hears a bit at time $\tau$ for the first time sets its level to $h(u) = \tau - t$ and forwards any bit in the following step. Since the nodes of the network know a constant factor upper bound on diameter $D$, after $O(D)$ time slots each node $u$ is aware of its distance $h(u)$ from the source $s$.

At the end of this phase, each node is aware of its level and can compute its subnetwork accordingly.

2.3 The Mcfs primitive

Consider two consecutive layers $\mathcal{L}(j)$ and $\mathcal{L}(j+\varepsilon)$, where $\varepsilon \in \{-1, +1\}$. We say that a subset of nodes $S \subseteq \mathcal{L}(j)$ has the collision-freeness property with respect to a subset of nodes $R \subseteq \mathcal{L}(j+\varepsilon)$ if no two nodes in $S$ share the same neighbor in $R$. Such a subset is said maximal if no node can be added to it without violating the collision-freeness property.

Our Mcfs protocol shares some features with the well-studied Maximal Independent Set (MIS) computation algorithm [1] [17] [18] and with the *Decay* protocol [3]. In fact, from the induced subgraph $g = (V(g) = R \cup S, E(g))$ consider the special subgraph $g'$ built as follows:

- $V(g') = S$ and
- an edge of $g'$ between $S_i, S_j \in S$ exists if and only if in the graph $g$, $S_i$ and $S_j$ have at least a common neighbor in $R$.

First, we remark that the set of senders $S = \{S_1, S_2, \cdots\}$ with respect to the receivers $R = \{R_1, R_2, \cdots\}$ with the maximal collision-freeness property corresponds to a maximal independent set of the graph $g'$ constructed from $g = (V(g) = R \cup S, E(g))$ as described above.

Next, during the whole execution of the MIS procedure (see for instance [1] or [17]), each node of the considered network should be aware of their current degrees in order to attempt to broadcast with probability inversely proportional to its degree which is difficult to implement in the context of radio networks. To get rid of this difficulty, we design an algorithm where each node attempts to access the channel with decreasing probabilities $\frac{1}{2}, \frac{1}{2^2}, \cdots, \frac{1}{2^\Omega(\log n)}$, so that we do not have to compute constantly the degrees of the still participating nodes.

With these remarks in mind, the following algorithm computes a subset of $S$ (first argument) with the maximal collision-freeness property w.r.t. $R$ (second argument).
Algorithm $\text{Mcfs}(S, R)$

**Input**: A set of senders $S$, a set of receivers $R$.

**Output**: Each $s \in S$ with a status $\in \{\text{INMcfs}, \text{NOTINMcfs}\}$

1. for $j$ from 0 to $c \log n$
   
   /* Each sender $s$ attempts to send its ID to the receivers */
   
   2. for $i$ from 0 to $c \log n$
      
      /* for each $s \in S$, send a message containing ID($s$) with probability $\frac{1}{2^i}$ */
      
      3. each receiver $r \in R$ that has correctly received a message marks itself
   
   /* The marked receivers attempt to acknowledge one of its received messages */
   
   4. for $i$ from 0 to $c \log n$
      
      /* 4 consecutive time slots are needed. */
      
      5. (i) for each marked $r \in R$, select u.a.r. one of the received IDs and with probability $\frac{1}{2^i}$, send it back.
      
      6. (ii) for each $s \in S$, upon the correct reception of its own ID: $s$ sends it back and changes its status to $\text{INMcfs}$.
      
      7. (iii) each $r \in R$ that hears noise (receiving collision) or a single messageprovokes collisions at the senders level (by sending intentionally a bit) in order to eliminate some participants.
      
      8. (iv) each $s \in S$ without the status $\text{INMcfs}$ that hears noise or a single message from the previous time slot sets its status to $\text{NOTINMcfs}$.

11. end
   
   // Decided nodes are eliminated.
   
   12. Each node in $S$ with fixed status does not participate to the next loop.
   
   13. Each marked node in $R$ removes its mark.

14. end

**Algorithm 1**: The $\text{Mcfs}$ algorithm.

To fix ideas, let us consider two consecutive layers $L(j)$ and $L(j+1)$ and suppose that we want to compute the $\text{Mcfs}$ in $L(j)$. It is easily seen that the algorithm needs $O(\log^2 n)$ time slots. We prove in the following lemma that it computes a maximal collision-free set w.h.p.

**Lemma 1** Fix $j \geq 1$. Let $S$ be a subset of the layer $L(j)$ and $R$ be a non-empty subset of the layer $L(j+1)$. There exists a constant $c$ corresponding to the one in the lines 1, 2 and 6 of the above algorithm such that the invocation of $\text{Mcfs}(S, R)$ computes $\{S_1, S_2, \cdots \} \subseteq S$ with the maximal collision-freeness property w.r.t. the set of nodes $R$ with probability at least $1 - O(\frac{1}{n^2})$.

**Proof.** The proof is inspired from those of $\text{Decay}$ [3] and $\text{Mis}$ [17]. In fact, if at each step we know the degree $d_v$ of a node $v$, a natural choice of probability in order to send a message to $v$ is $O(\frac{1}{d_v})$ but as nodes and edges are removed iteration after iteration it is difficult in the context of radio networks to maintain or approximate accurately the degrees of the still participating nodes. Thus, as in $\text{Decay}$ we use the decreasing probabilities $2^{-i}$ to cope with these conditions.

Let $r$ be a node of $R$ with $d_r$ neighbors $s_1, \cdots s_{d_r} \in S$. Let $p_r$ be the probability that $r$ receives a message from at least one of its neighbors during the $c \log n$ time slots of the
first inner for loop of the algorithm (lines 2 — 5). As a main properties of \textsc{decay}, it has been proved in [13] that there exists a constant \(c\) such that \(p_r\) is greater than \(\frac{1}{2}\). Thus, \(r\) is marked with probability at least \(\frac{1}{2}\). As the second loop has the same behaviour, we claim that \(r\) succeeds to send back the ID of one of its neighbors \(s_j\) with probability at least \(\frac{1}{4}\). The proof is now similar to the one of the MIS algorithm [17] (for instance, see the proof of [18 Corollary 1]). The edge \((r, s_j)\) is removed from the algorithm (as the status of \(s_j\) is fixed) with probability at least \(\frac{1}{4}\). The expected number of edges removed from the remaining graph is then at least \(1/4\) of the previous number of edges at any phase of the outer loop. Hence, after \(r\) iterations of the outer loop the expected number of remaining edges is less than \(\frac{n^2}{2^{r+1}}\). By choosing \(r = c \log n\), for some sufficiently large constant \(c\), after \(r\) steps the probability that any edge between \(S\) and \(R\) remains is less than \(O\left(\frac{1}{n^r}\right)\).

\[ \square \]

2.4 Computing the skeleton trees

Given a subnetwork \(\mathcal{J}_i\) with the set of consecutive layers \(\mathcal{L}((i - 1)\lambda), \mathcal{L}((i - 1)\lambda + 1), \cdots, \mathcal{L}(i\lambda - 1)\), we construct its skeleton trees starting from the leaves down to the roots. For sake of simplicity and clarity, we suppose that the procedure \textsc{mcfs}(A, B) returns the subset of \(A\) that has the maximal collision-free property w.r.t. \(B\) so that we can write

\[ A' = \text{mcfs}(A, B). \]

For example, in the previous figure the \textsc{mcfs} procedure finds (layers \(\mathcal{L}(7)\) and \(\mathcal{L}(8)\))

\[ \{2\} = \text{mcfs}([2, 8, 9], \{6, 17\}). \]

The following algorithm computes the skeleton trees of the subnetwork \(\mathcal{J}_i\).

\begin{algorithm}
\textbf{Algorithm} \textsc{skeletonTrees}(i)
\begin{algorithmic}
\STATE \textbf{Input} : An integer \(i\) which corresponds to the subnetwork \(\mathcal{J}_i\).
\STATE \textbf{Output}: The skeleton trees of \(\mathcal{J}_i\) (each node of \(\mathcal{J}_i\) knowing if it is in the skeleton trees or not).
\STATE \hspace{2em} // Initialization intended for the gluing process
\STATE \hspace{3em} \(\mathcal{L}'(i\lambda - 1) := \text{mcfs}(\mathcal{L}(i\lambda - 1), \mathcal{L}(i\lambda - 2))\) \hspace{1em} // From the leaves down to the roots
\STATE \hspace{2em} \textbf{for} \(k\) from \(i\lambda - 2\) down to \((i - 1)\lambda\) \textbf{do}
\STATE \hspace{3em} \(\mathcal{L}'(k) := \text{mcfs}(\mathcal{L}(k), \mathcal{L}'(k + 1))\)
\STATE \hspace{2em} \textbf{end}
\end{algorithmic}
\end{algorithm}

Algorithm 2: Computing the skeleton trees of \(\mathcal{J}_i\).

Let us follow the execution of \textsc{skeletonTrees}(2) with the example of the Figure 2. We have successively

\[ \mathcal{L}'(9) = \text{mcfs}(\mathcal{L}(9), \mathcal{L}(8)) = \text{mcfs}([13, 15, 4, 1, 16, 5], \{17, 21, 3, 6, 22, 11\}). \]

It finds for example \(\mathcal{L}'(9) = \{15, 16\}\). The computation of \(\mathcal{L}'(8) = \text{mcfs}(\mathcal{L}(8), \{15, 16\})\) gives \(\{6, 17\}\). Then \(\mathcal{L}'(7) = \text{mcfs}(\mathcal{L}(7), \{6, 17\}) = \{2\}\), \(\mathcal{L}'(6) = \text{mcfs}(\mathcal{L}(6), \{2\}) = \{10\}\) and \(\mathcal{L}'(5) = \text{mcfs}(\mathcal{L}(5), \{10\}) = \{14\}\).

\begin{lemma}
For any \(i \in \{1, 2, \cdots, \mathbb{P}\}\), the invocation of \textsc{skeletonTrees}(i) computes the skeleton trees (as defined in ??) of the subnetwork \(\mathcal{J}_i\) w.h.p.
\end{lemma}
Proof. By Lemma 1, the initialization (line 1 of the algorithm) computes w.h.p. a set with the maximal collision-freeness property in the last layer $L(i\lambda - 1)$ w.r.t. to the layer $L(i\lambda - 2)$. The same process is applied for each $j \in [i\lambda - 2, (i - 1)\lambda]$. It is easily seen that $\text{Card}(L'(j+1)) \geq \text{Card}(L'(j))$ and each node of $L'(j+1)$ is connected to at most one node of $L'(j)$ (otherwise the collision-free property is violated), we then build a set of trees. Since the process is repeated a constant number of times inside the for loop ($\lambda = O(1)$), and as we can choose the constant of Lemma 1 to get probabilities of at least $1 - O(1/\alpha^2)$ for each call to the procedure $\text{Mcfs}$, the proof of the assertion “w.h.p.” in the Lemma is now trivial. □

2.5 The zig-zag gluing process

The skeleton trees of the subnetworks can be constructed in parallel (between the $D/\lambda$ subnetworks) but once built, we need to connect the leaves of the skeleton trees of $\mathcal{S}_i$ to the roots of those of $\mathcal{S}_{i+1}$ (they are not necessarily direct neighbors). To do so, we need to schedule a way to send a message arriving at the set $L'(i\lambda - 1)$ (the leaves of $\mathcal{S}_i$’s trees computed with the $\text{SkeletonTrees}(i)$ algorithm) to the set $L'(i\lambda)$ (the roots of $\mathcal{S}_{i+1}$’s trees).

The solution consists of the construction of two new subsets denoted $\mathcal{G}(i\lambda - 2)$ and $\mathcal{G}(i\lambda - 1)$ respectively inside the 2 highest level layers of $\mathcal{S}_i$, viz. $L(i\lambda - 2)$ and $L(i\lambda - 1)$. In fact, by setting $\mathcal{G}(i\lambda - 1) = \text{Mcfs}(L(i\lambda - 1), L'(i\lambda))$ and $\mathcal{G}(i\lambda - 2) = \text{Mcfs}(L(i\lambda - 2), \mathcal{G}(i\lambda - 1))$ we remark that a message arriving at nodes of $L'(i\lambda - 2)$ can be forwarded without collision in one step to the nodes of $L'(i\lambda - 1)$, then it can be sent back still without collision to the nodes of $\mathcal{G}(i\lambda - 2)$ in the next time slot, then forwarded successively to the nodes of $\mathcal{G}(i\lambda - 1)$ and finally to the roots of $L'(i\lambda)$. We call this stitch process : the zig-zag gluing process (forward, back and forward twice).

The following figures illustrate such a mechanism:

![Figure 3. The zig-zag process diagram.](image1.png)

![Figure 4. An example of the process at the frontier of $\mathcal{S}_2$ and $\mathcal{S}_3$. $\mathcal{G}(8), \mathcal{G}(9)$ as well as their links are in dashed blue.](image2.png)

In Figure 4, $\{48, 66\}$ are the roots of $\mathcal{S}_3$. We have $\mathcal{G}(9) = \text{Mcfs}(L(9), \{48, 66\}) = \{13\}$. $\mathcal{G}(8) = \text{Mcfs}(L(8), \{13\}) = \{3\}$. When $M$ arrives at 6 and 17, it is forwarded to
{15, 16} and sent back to {3}. Then it is forwarded to {13} and then immediately to the roots {48, 66} of $\mathcal{F}_3$. Thus in this example, we have connected the subnetworks $\mathcal{F}_2$ and $\mathcal{F}_3$.

We have the following crucial lemma.

**Lemma 3** Suppose that the skeleton trees of the subnetworks $\mathcal{F}_i$ are built ($1 \leq i \leq \frac{D}{\lambda}$). After the invocations of

$$G(i\lambda - 1) = \text{MCFS}(\mathcal{L}(i\lambda - 1), \mathcal{L}'(i\lambda))$$

and

$$G(i\lambda - 2) = \text{MCFS}(\mathcal{L}(i\lambda - 2), G(i\lambda - 1))$$

then w.h.p. there exists a collision-free schedule such that when the message $M$ arrives at nodes of $\mathcal{L}'(i\lambda - 1)$, it can be sent back to some nodes of $G(i\lambda - 2)$ in the next time slot, then forwarded during the two next steps successively to some nodes of $G(i\lambda - 1)$ and finally to all the roots of $\mathcal{L}'(i\lambda)$.

**Proof.** The “w.h.p.” assertion is inherently linked to the probability that the $\text{MCFS}$ algorithm succeeds.

First, the procedure of equation (??) builds a one step collision-free schedule between $G(i\lambda - 1)$ and $\mathcal{L}'(i\lambda)$: by the computation of $G(i\lambda - 1)$, in one step all the roots of $\mathcal{L}'(i\lambda)$ can be reached without interference from $G(i\lambda - 1)$. The same observation applies to $G(i\lambda - 2)$ and $G(i\lambda - 1)$ after the invocation of (??).

Next, we have to prove that there is always some collision-free connections from $\mathcal{L}'(i\lambda - 1)$ to $G(i\lambda - 2)$ following the calls (??) and (??).

There should be some edges between these two sets otherwise $\mathcal{L}'(i\lambda - 1)$ is not maximal but since $\mathcal{L}'(i\lambda - 1)$ is also collision-free w.r.t. the whole layer $\mathcal{L}(i\lambda - 2)$ (by construction as done by the line 1 of the $\text{SKELETONTREES}$ algorithm), two nodes of $\mathcal{L}'(i\lambda - 1)$ can’t share the same neighbor in the whole layer $\mathcal{L}(i\lambda - 2)$ and then, a fortiori, in $G(i\lambda - 2)$. Thus, all the existing links from $\mathcal{L}'(i\lambda - 1)$ to $G(i\lambda - 2)$ are collision-free.

\[ \square \]

### 2.6 Putting it all together

Having defined all of the primitives needed, we are now ready to summarize our main algorithm as follows:

**Algorithm** $\text{MAIN}()$

1. Layer the network (using $O(D)$ time slots).

2. for $i \in \{1, 2 \cdots, \frac{D}{\lambda}\}$ in parallel do do

3. Use $O(\log^2 n)$ time slots to compute the skeleton trees of $\mathcal{F}_i$.

4. Use $O(\log^2 n)$ time slots and compute

5. $G(i\lambda - 1)$ and $G(i\lambda - 2)$ using (??) and (??).

6. end

7. From the source $s$, send $M$ through the skeleton trees (use the zig-zag process at the frontiers of the subnetworks). This takes $O(D)$ time slots.

8. For each active node, flood the network with the message $M$ invoking the $\text{DECAY}$ protocol for $O(1)$ times.

**Algorithm 3:** Sketch of the $\text{MAIN}$ procedure.
Proof of Theorem 1. To prove our result, we need to prove that all the nodes of the network are at constant hop-distance of the nodes of the skeleton trees so that the line 8 of the MAIN algorithm suffices to broadcast $M$ to the still inactive nodes.

Let $v$ be a node in the subnetwork $\mathcal{S}_i$ ($i > 1$) and suppose that $v$ is not a node of the skeleton trees. Suppose that $h(v) = j$.

If $j = i\lambda - 1$ then $v$ is at distance $\leq 2$ of the leaves of the skeleton trees.

If $j = i\lambda - 2$ then we have two sub-cases. If $v$ has at least one neighbor in the layer $\mathcal{L}(i\lambda - 1)$ then it is exactly at distance 1 of a leaf of the skeleton trees. If $v$ has no neighbors in $\mathcal{L}(i\lambda - 1)$, we argue that there is a path of length at most $\lambda$ from the layer $\mathcal{L}((i - 1)\lambda - 2)$ to $v$ in the previous subnetwork $\mathcal{S}_{i-1}$. $v$ is directly connected via this path to the leaves of the trees of $\mathcal{S}_{i-1}$ or to a node $u$ of the layer $\mathcal{L}((i - 1)\lambda - 2)$. In both cases, $v$ is at constant distance from a node which belongs to the skeleton trees.

Similar arguments hold if $v$ belongs to the previous layers ($j \leq i\lambda - 3$).

3 Conclusion

As archetypal methods in multi-hop radio networks and in distributed computing, the MIS algorithm ([17]) as well as the DECAY protocol ([3]) have already greatly profit to many researchers. Like many others before us, we also profit of their design and analyses in order to provide a relatively simple and elegant solution to a long unresolved research problem ([2]) about radio networks with collision detection: we have proved that there exists a randomized algorithm that solves the broadcasting problem in time $O(D + \log^2 n)$ with high probability. According to the well-known lower-bounds $\Omega(\log^2 n)$ and $\Omega(D)$ in this context, our proposed algorithm is optimal. Our main contribution is the development of two fundamental primitives: (i) MCFS that allows to build the skeleton trees of the network and (ii) a stitching process that merges these backbones together leading to a collision-free schedule for broadcasting algorithms. These two primitives are of independent interest on their own.

References

[1] Alon, N. and Babai, L. and Itai, A. (1986) A fast and simple randomized parallel algorithm for the maximal independent set problem. Journal of Algorithms, 7, 567 – 583.

[2] Alon, N. and Bar-Noy, A. and Linial, N. and Peleg, D. (1991) A Lower Bound for Radio Broadcast. Journal of Computer and System Sciences 43, 290 – 298.

[3] Bar-Yehuda, R. and Goldreich, O. and Itai, A. (1992) 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, 104 – 126.

[4] Bar-Yehuda, R. and Israeli, A. and Itai, A. (1993) Multiple communication in multi-hop radio networks. SIAM Journal on Computing 22, 875 – 887.

[5] Chlamtac, I. and Kutten., S. (1985) On broadcasting in radio networks: Problem analysis and protocol design. IEEE Transactions on Communications 33, 1240 – 1246.
[6] Chlebus, B. S. and Kowalski, D. R. and Pelc, A. and Rokicki, M. A. (2011) Efficient distributed communication in ad-hoc radio networks. In Proc. ICALP’11, 613 – 624.

[7] Czumaj, A. and Rytter, W. (2006) Broadcasting algorithms in radio networks with unknown topology. Journal of Algorithms 60 115 – 143. Extended abstract in Proc. FOCS’03: 492 – 501.

[8] Gasieniec, L. and Peleg, D., and Xin, Q. (2005) Faster communication in known topology radio networks. In Proc. PODC’05: 129 – 137.

[9] Ghaffari, M. and Haeupler, B. (2013) Near Optimal Leader Election in Multi-Hop Radio Networks. In Proc. SODA’13: 748 – 766.

[10] Ghaffari, M. and Haeupler, B. and Khabbazian, M. (2015) Randomized broadcast in radio networks with collision detection. Distributed Computing 28 407 – 422 Extended abstract in Proc. PODC’13: 325 – 334.

[11] Greenberg, A. G. and Flajolet, P. and Ladner, E. R. (1987) Estimating the Multiplicities of Conflicts to Speed Their Resolution in Multiple Access Channels. Journal of the ACM 34 289 – 325.

[12] Haeupler, B. and Wajc, D. (2016) A Faster Distributed Radio Broadcast Primitive: Extended Abstract. In Proc. PODC’16: 361 – 370.

[13] Khabbazian, M. and Kowalski, D. (2011) Time-efficient randomized multiple-message broadcast in radio networks. In Proc. PODC’11: 373 – 380.

[14] Kowalski, D., and Pelc, A. (2003) Broadcasting in undirected ad hoc radio networks. In Proc. PODC’03: 73 – 82.

[15] Kowalski, D. R., and Pelc, A. (2007) Optimal deterministic broadcasting in known topology radio networks. Distributed Computing 19, 185 – 195.

[16] Kushilevitz, E., and Mansour, Y. (1993) An Ω(D log (N/D)) lower bound for broadcast in radio networks. In Proc. PODC’93: 65 – 74.

[17] Luby, M. (1986) A Simple Parallel Algorithm for the Maximal Independent Set Problem. SIAM J. Comput. 15, 1036–1053.

[18] Métivier, Y. and Robson, J. M. and Saheb-Djahromi, N. and Zemmari, A. (2011) An optimal bit complexity randomized distributed MIS algorithm. Distributed Computing 23, 331 – 340.

[19] Peleg, D. (2007) Time-efficient broadcasting in radio networks: A review. In Proc. International Conference on Distributed Computing and Internet Technologies’07: 1 – 18.

[20] Willard, D. E. (1986) Log-Logarithmic Selection Resolution Protocols in a Multiple Access Channel SIAM Journal on Computing 15 468 – 477.