Εθνικο Μετσοβιο Πολυτεχνειο
Τμημα Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Λογικής και Επιστήμης Υπολογιστών CoReLab

Επικοινωνία ένας προς όλους και όλοι προς όλους σε Δίκτυα Ραδιοεκπομπής

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Παράσχου Ε. Κουτρή

Επιβλέπων: Ευστάθιος Ζάχος
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2009

Εθνικο Μετσοβιο Πολυτεχνειο
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙ-
ΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Λογικής και Επιστήμης Υπολογιστών CoReLab

Επικοινωνία ένας προς όλους και όλοι προς όλους σε Δίκτυα Ραδιοεκπομπής

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Παράσχου Ε. Κουτρή

Επιβλέπων: Ευστάθιος Ζάχος
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την $22^{η}$ Ιουλίου 2009.

....	....	....
Ευστάθιος Ζάχος	Άρης Παγουρτζής	Δημήτριος Φωτάκης
Καθηγητής Ε.Μ.Π.	Λέκτορας Ε.Μ.Π.	Λέκτορας Ε.Μ.Π.

Αθήνα, Ιούλιος 2009

....
Παράσχος Ε. Κουτρής
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Σκοπός της διπλωματικής αυτής είναι η θεωρητική μελέτη ενός μοντέλου επικοινωνίας που ονομάζεται *δίκτυο ραδιοεκπομπής*. Σε ένα τέτοιο δίκτυο, θεωρούμε ότι το μήνυμα που μεταδίδει ένας κόμβος λαμβάνεται από όλους τους γειτονικούς κόμβους. Σε περίπτωση, όμως, που δύο ή περισσότεροι γειτονικοί κόμβοι μεταδίδουν ταυτόχρονα, συμβαίνει σύγκρουση και το μήνυμα δεν φτάνει στον προορισμό του. Το χαρακτηριστικό αυτό προσδίδει στο μοντέλο των δικτύων ραδιοεκπομπής μια επιπλέον δυσκολία σε σύγκριση με άλλα μοντέλα επικοινωνίας. Επίσης, θεωρούμε ότι η επικοινωνία πραγματοποιείται σε συγκεκριμένες χρονικές στιγμές. Με βάση αυτό το μοντέλο, εξετάζουμε δύο βασικές διεργασίες: την *επικοινωνία ένας προς όλους*, όπου το μήνυμα που βρίσκεται σε ένα συγκεκριμένο κόμβο πρέπει να διανεμηθεί σε όλους τους κόμβους του δικτύου, και την *επικοινωνία όλοι προς όλους*, όπου ο κάθε κόμβος διαθέτει ένα μήνυμα και κάθε μήνυμα πρέπει να φτάσει σε όλους τους κόμβους. Θα μελετήσουμε διάφορες παραλλαγές του προβλήματος αυτού, ανάλογα με παραμέτρους όπως τη γνώση των κόμβων για την τοπολογία του δικτύου, την ύπαρξη μηχανισμού εντοπισμού συγκρούσεων, την τοπολογία του δικτύου, τον τρόπο που οι κόμβοι ενεργοποιούνται και τη χρήση τυχαιότητας. Για κάθε παραλλαγή, θα παρουσιάσουμε πρωτόκολλα που συναντήσαμε στη βιβλιογραφία, καθώς και αποτελέσματα για κάτω φράγματα στην επίδοση των αλγορίθμων. Τέλος, θα μελετήσουμε ένα συνδυαστικό μαθηματικό εργαλείο που χρησιμοποιείται σε πρωτόκολλα στα δίκτυα ραδιοεκπομπής, τους selectors.

Abstract

The purpose of this diploma thesis is the theoretic study of the communication model of a *radio network*. In a radio network, once a node transmits, all the neighboring nodes receive the message; however, in the case that two or more neighbors of a node transmit simultaneously, a collision occurs and the message is lost. This characteristic makes communication a much more demanding task compared to other communication models. We also make the assumption that the communication is performed in synchronous rounds. Under this model, we study two basic communication tasks: *broadcasting*, where a message initially held by a source node has to be disseminated to all other nodes in the network and *gossiping*, where each node holds a distinct message and each message has to be distributed to all other nodes. We consider several variants of the problem, which depend on parameters such as the accessibility of knowledge about the network topology, the availability of collision detection mechanisms, the wake-up model, the topology of the network and the use of randomization. For each variant, we present protocols that have been proposed in the literature, as well as lower bounds and impossibility results. Finally, we study a combinatorial tool essential for protocols in radio networks, the *selectors*.

Keywords

radio network, broadcasting, gossiping, ad-hoc network, selector, selective family, collision, fault-tolerance, geometric network, randomization

Ευχαριστίες

Με την ολοκλήρωση της εκπόνησης της διπλωματικής μου εργασίας, θα ήθελα να απευθύνω τις ευχαριστίες μου σε όλους τους ανθρώπους που με βοήθησαν και με συνόδεψαν αυτά τα πέντε χρόνια των σπουδών μου στο Πολυτεχνείο.

Καταρχάς, στον καθηγητή κ. Ε. Ζάχο, που από το πρώτο εξάμηνο σπουδών μου στη σχολή, μου έδωσε τροφή για σκέψη και με ώθησε στο να αγαπήσω τη Θεωρητική Πληροφορική και να ασχοληθώ σε μεγαλύτερο βάθος μαζί της. Στο λέκτορα κ. Α. Παγουρτζή, που μου έδωσε το έναυσμα για το θέμα της διπλωματικής αυτής εργασίας και στη συνέχεια με βοήθησε καθόλη τη διάρκειά της. Στο λέκτορα κ. Δ. Φωτάκη για το ενδιαφέρον που έδειξε και τις χρήσιμες συμβουλές και παρατηρήσεις του. Επίσης, θα ήθελα να ευχαριστήσω τον κ. Ν. Παπασπύρου και κ. Κ. Σαγώνα, που σε όλα τα χρόνια των σπουδών μου, υπήρξαν καλοί δάσκαλοι για μένα.

Ακόμα, ευχαριστώ από την καρδιά μου τους φίλους-συμφοιτητές μου για τις ωραίες στιγμές που ζήσαμε όλα αυτά τα χρόνια, την όμορφη συνεργασία και τα εποικοδομητικά ξενύχτια μας, καθώς και για όλα εκείνα τα κοινά που περάσαμε μαζί και θα θυμάμαι έντονα από τα χρόνια των σπουδών μου.

Τέλος, θα ήθελα να πω ένα μεγάλο ευχαριστώ στην οικογένειά μου και ιδιαίτερα στους γονείς μου για την στήριξή τους και τις πολύτιμες συμβουλές που μου προσέφεραν μέχρι σήμερα.

Contents

List of Figures

List of Algorithms

Chapter 1

The model

1.1 Introduction

Over the years, many communication models have been proposed for different kind of physical networks. Each model focuses on a different combination of characteristics of the network we are interested in. The network model we study here is not the common network with *point-to-point* channels, but a network that uses *multi-access* channels for the communication (fig. 1.1). A multi-access channel has two fundamental characteristics: it can can be accessed by many nodes and a signal reaches all the nodes connected to the channel only in the case it is the only signal occupying the channel at that time. If two or more signals are transmitted through the same channel simultaneously, a collision occurs and none of the signals is delivered, as the collision creates noise. This characteristic consists the motivation behind the model of the *radio network*, which is the theme of this thesis.

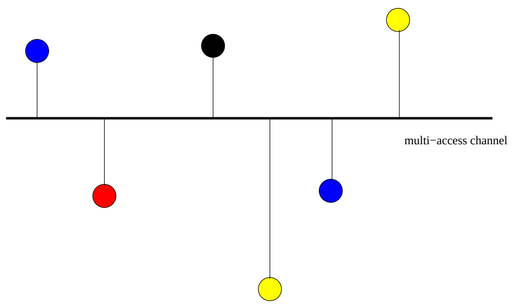


Figure 1.1: A multi-access channel

A *radio network* is a collection of devices that can either transmit or receive messages. We will refer to these devices as *nodes*. Each node can reach only a subset of the other nodes, depending on the power of its transmitter and on the topological characteristics of the area where the network is located. For example, obstacles such as walls, buildings or natural objects can prevent a node which would otherwise be in the proximity of the node transmitting to receive. When every node of the network can reach any other node, we refer to the network as a *single-hop* radio network. Else, we have a *multi-hop* radio network. In such a network, communication between nodes that are not within reach is performed by consecutive transmissions.

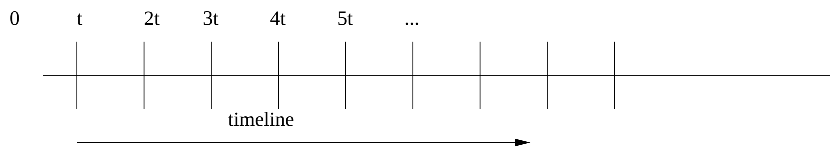


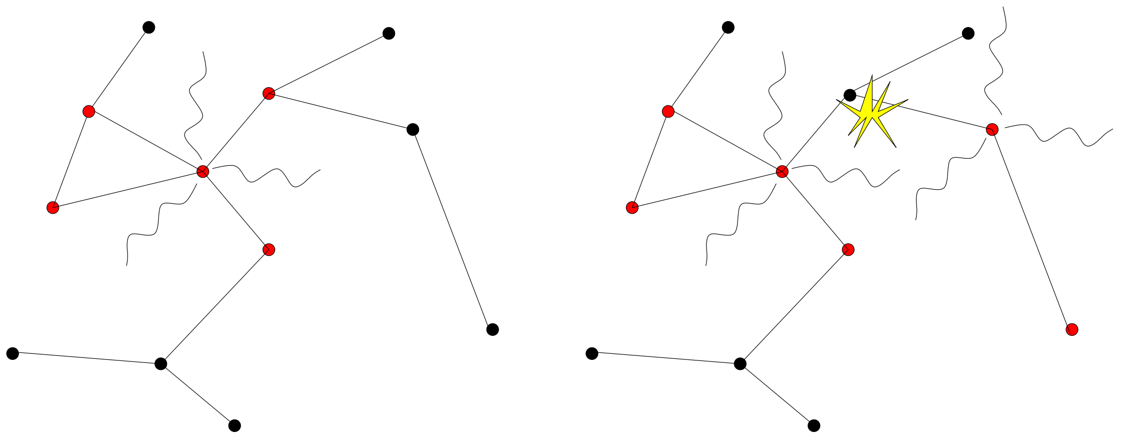
Figure 1.2: A node of a radio network transmits only in specific time-slots.

A radio network is commonly modeled as a graph, where a node corresponds to a node of the network and an edge (u, v) exists when node u can reach node v. We call v a *neighbor* of node u. Communication in radio networks is structured in synchronous *rounds*, i.e. a node can only transmit in specific time-slots (figure 1.2). In every round, each node acts either as a transmitter or as a receiver. When a node u acts as a transmitter, the message transmitted is sent to every neighboring node of u. On the contrary, a node v acting as a receiver collects a message if and only if exactly one of its neighbors transmits a message at the same round. If two or more neighbors of v transmit at the same round, then a *collision* occurs at v and the message does not reach v. Both cases are depicted in figure 1.3.

The above characteristics of communication in radio networks consist the main difficulty in designing algorithms that perform communication tasks. One might think at first that transmitting simultaneously a message to all the neighbors means more efficient communication, however, the collisions that may occur slow down the communication speed. Consider a node with many neighbors; the neighboring nodes must be synchronized so that only one of them transmits. This inherent difficulty of the model makes the design of fast communication algorithms in radio networks more demanding than in most network models.

1.2 Graph Models

As we have mentioned before, a radio network is modeled as a graph. We will mostly deal with directed graphs, which is the most general model. Nevertheless, we will often consider more restricted network topologies, where more efficient algorithms can be designed. Generally, we have the following categories:



(a) Transmission without collision: the message reaches all neighboring nodes.

(b) Transmission with collision: the node with two neighboring nodes acting as transmitters hears only noise.

Figure 1.3: Successful transmission and collision.

Symmetric Graphs. A logical assumption we can make for the underlying graph is that it is *symmetric*. Specifically, a node u can reach some other node v if and only if v can reach u. A symmetric graph can be thought as an *undirected* graph, so from now on these two models will be considered identical.

Geometric Graphs. *Geometric graphs (GRN)* describe a very natural structure of a network. Each node is represented by a point in the plane and is assigned a *transmission range*. A node can reach only the nodes within its own range. In this model, we can consider two subcategories: each node has its own range or the ranges are uniform.

Special Topologies. We will also deal with specific graph topologies, such as *lines, hexagonal* or *square grids* and *rings.*

Let us also introduce some important parameters of the underlying graph, which affect the performance of the algorithms for any communication task.

n : The number of nodes in the graph (or size of the graph).

D : If we have specified some central (or source) node, D denotes the *eccentricity* of that node, i.e. the maximum distance from the node to any other node of the graph. Otherwise, D denotes the *maximum eccentricity* of the graph.

Δ : The maximum *in-degree* (number of in-neighbors) of the graph.

1.3 Communication Scenarios

Although the communication in radio networks has some fundamental properties, we can make various assumptions about the characteristics of the communication process. These assumptions lead to different models and affect the design and efficiency of the algorithms. We will present here the basic choices one has to make in order to specify a communication scenario.

Centralized / Distributed. A basic distinction for networks is whether there is some centralized control or the network is distributed. In the setting of radio networks, a centralized algorithm means that the nodes have complete knowledge of the topology of the network. Thus, any algorithm can exploit information about the topology of the underlying graph. However, centralized control is not always a realistic assumption, in view of the dynamical nature of modern networks: topologies can be frequently changing, nodes might be removed or added to the network. In such cases, the model of *ad-hoc radio networks*, where each node is aware only of its own ID (and possibly the number of nodes), is more suitable. Another logical assumption we could make is that the nodes are only aware of their local neighborhood and not the whole network.

Randomization / Determinism. It is also crucial to decide whether we can use randomization in our algorithms. Randomization, in contrary to determinism, does not guarantee that the communication task will be always completed, but performs the task with high probability. However, randomized algorithms perform faster and are easily implemented in a distributed setting. Specifically in the setting of radio networks, randomization has proved to be very powerful.

Adaptiveness / Obliviousness. Another feature that affects communication is adaptiveness. We call an algorithm *adaptive* when a node decides for the next transmission based on its previous history, i.e. the messages it has so far received. On the other hand, *oblivious* algorithms schedule the transmissions in a predefined way. Clearly, an adaptive algorithm exploits information gathered from neighboring nodes in order to transmit in a more efficient way. On the other hand, an oblivious algorithm is in most cases quite simple and easily implementable.

Collision Detection. When a collision happens, noise is created and no message is delivered. Nevertheless, we can distinguish two cases. In the first case, where we assume *collision detection*, the noise created from the collision is different from the background noise and, as a result, the node can distinguish a collision from an absence of transmission. In the second case, the two events are indistinguishable. We will mostly deal with the second case, which consists the more harsh model.

Fault tolerance. Assuming that a network functions without any faults is not a realistic assumption, as failures are frequent nowadays. Thus, we should consider the fact that during the execution of the algorithm, a component of the network may stop functioning or start behaving maliciously. *Fault tolerant* algorithms guarantee that the communication task will be completed for every fault-free node. Failures can be of various types: loss of messages, Byzantine failures (a node may transmit an arbitrary message), node failures.

1.4 Communication Tasks

We will study the two fundamental communication tasks of any network, *broadcasting* and *gossiping.*

Broadcasting (or *one-to-all* communication) is the task of communicating a message from a source node to all the nodes of the radio network. Clearly, broadcasting can help to transfer some information over the whole network or even be used as a subroutine for more complex communication tasks.

Gossiping (or *all-to-all* communication) assumes that each node has a distinct message and the goal is to disseminate all messages to all the nodes.

Another communication task that has been studied in the setting of radio networks is *leader election*, where the nodes must agree on some node of the network as the leader. There has also been considered a generalization of the gossiping problem, the *multicast* problem, where the goal is to exchange messages within a specific set of nodes of the network. Finally, another problem which has recently received attention is the *wake-up* problem, where the task consists of waking up a collection of processors connected by a multihop ad hoc radio network with no access to a global clock.

Chapter 2

Selectors

2.1 Introduction

A basic combinatorial tool used extensively in algorithms for broadcasting and gossiping in ad-hoc radio networks are *selectors*. A *selector* is a notion that generalizes a wide group of several combinatorial tools, such as *k-selectors*, *k-selective families*, *strongly selective families*, *superimposed codes* and *cover-free families*. These tools have not only applications to broadcasting and gossiping protocols for ad-hoc radio networks, but also to other areas, such as deterministic conflict resolution in multiple-access channels, deterministic algorithms for group testing and other communication tasks concerning radio networks (the wake-up problem, leader election, local clock synchronization). In addition, many special cases of selectors have been motivated by purely combinatorial interests. Since selectors play a crucial role in distributed algorithms for radio networks, this section will be devoted to a presentation of the theory behind. An interesting survey for selectors can be found in [16].

The key concept behind selectors is *selection by intersection*. By this, we mean that we distinguish a single element of a set as the only common element between this set and some other set. In other terms, given a subset $X \subseteq U$ of a finite domain U, an element $z \in X$ is *selected* by a set $Y \subseteq U$ when $|Y \cap X| = \{z\}$. We equivalently say that Y *hits* X at z.

The next step is to examine the way we can parametrize the selection properties of such structures. The first thing we consider is the size n of the finite domain U from where we choose the elements. Then, we are interested in the size of the subsets we want to hit. We can impose a restriction on this size and consider only subsets of size less than some parameter k. This leads us to the definition of a *k-selective family* by Chlebus et al. [15].

Definition 2.1.1. *A family $\mathcal{S} = \{S_1, \dots, S_t\}$ of subsets of a set U is k-**selective** for U if for any nonempty $X \subseteq U$ such that $|X| \leq k$, there exists a set $S_i \in \mathcal{S}$, such that $|X \cap S_i| = 1$.*

If the size n of the universe U is not clear from the context, we can use instead the term (n,k)-selective family. Another important category of selectors is the *k-selector*, which was introduced by Chrobak, Gasieniec and Rytter [18]. The basic difference is that we want to find some set X that not only hits some given set, but also *avoids* some other set, i.e. has no common elements with it. We thus have the following definition.

Definition 2.1.2. *Let $k, n (k \leq n)$ be positive integers. A family $\mathcal{S}$ of subsets of $[n] = \{1, 2, \ldots, n\}$ is called a **k-selector** (or (n,k)-selector) if for any two disjoint sets $X, Y \subseteq [n]$ with $k/2 \leq |X| \leq k$ and $|Y| \leq k$, there exists a set $S \in \mathcal{S}$ such that :*

- $|S \cap X| = 1$ *(S hits X)*
- $|S \cap Y| = 0$ *(S avoids Y)*

Now, we can think even further and ask for selectors that guarantee the selection of more than one element from a given set. Clearly, this consists a stronger combinatorial property. Following this thought, Clementi, Monti and Silvestri [23] introduced the notion of *strongly k-selective families.*

Definition 2.1.3. *Let $k, n (k \leq n)$ be positive integers. A family $\mathcal{S}$ of subsets of $[n] = \{1, 2, \ldots, n\}$ is called a **strongly k-selective family** (or strongly (n,k)-selective family) if for any nonempty set $X \subseteq [n]$ with $|X| \leq k$ and for every element $z \in X$, there exists a set $S \in \mathcal{S}$ such that $|S \cap X| = \{z\}$.*

The above combinatorial structures have some differences, but the concept behind is basically the same. Consequently, it is logical to ask whether there exists a general combinatorial structure which contains these notions. The answer is affirmative, as De Bonis, Gasieniec and Vaccaro [6] introduced the generalized notion of a (n,k,r)*-selector.*

Definition 2.1.4. *Let n, k, r be positive integers so that $r \leq k \leq n$. Let $\mathcal{S}$ be a family of subsets of $[n] = \{1, 2, \ldots, n\}$. We say that $\mathcal{S}$ is an (n,k,r)**-selector** if, for each set $X \subseteq [n]$ of size $|X| = k$, there are at least r elements in X that can be selected from X by sets in $\mathcal{S}$.*

We can now state the first definitions presented in terms of (n,k,r)-selectors.

- (n,k)-selective families are $(n,k,1)$-selectors.
- (n,k)-selectors correspond to $(n,2k,3k/2)$-selectors.
- strongly (n,k)-selective families are (n,k,k)-selectors.

Another extension of the notion of a k-selective family was presented in [42], where the authors defined *linearly k-selective families.* This combinatorial object can be viewed as a notion that lies between the selective and strongly selective families.

Definition 2.1.5. *A family $\mathcal{S}$ of subsets of $n = \{1, 2, \ldots, n\}$ is **linearly k-selective** for a positive integer k , if for any nonempty subset $X \subseteq [n]$ such that $|X| \leq k$, there is a set $Y \in \mathcal{S}$ that hits X on more than half of the elements in X.*

The advantage of linearly k-selective families is that, whereas they have a stronger combinatorial property compared to k-selectors, we can use k-selectors in order to obtain linearly k-selective families of asymptotically the same size.

Finally, we should also mention another selector with even stronger properties, the *path selector*, which was introduced in [45] and was used for gossiping.

2.2 Existence of selectors with small size

The task of constructing any selector that satisfies all the above properties is quite simple. For example, consider the family consisting only of singletons, i.e. the family $\{\{1\}, \{2\}, \ldots, \{n\}\}$. Clearly, this family hits any subset of $[n]$ on all the elements. However, the *size* of this family is n, whereas we would like to use selectors with as small size as possible. The question is whether we can construct selectors of small size, or even prove their existence. In this section, we will deal with the latter question.

The basic tool for proving the existence of small selectors is the *probabilistic method.* The probabilistic method is based on the following idea: we define a carefully chosen probability distribution over the combinatorial object and then prove that the probability that this object satisfies the desired properties is positive. Thus, there must be some instance of the combinatorial structure that holds the properties we want.

As far as k-selective families is concerned, Komlós and Greenberg [49] showed that there exist k-selective families of size $\mathcal{O}(k \log(n/k))$. Clementi, Monti and Silvestri [22] showed that any k-selective family must have size $\Omega(n \log(n/k))$. Thus, a k-selective family of size $\mathcal{O}(k \log(n/k))$ is optimal. The proof of the lower bound is based on the properties of another combinatorial tool, *intersection-free families.*

As for k-selectors, Chrobak et al. [18] showed with a simple probabilistic argument that there exist k-selectors of size $\mathcal{O}(k \log n)$. We will present the proof of this, in order to show how the probabilistic method is used.

Lemma 2.2.1. *For any positive integers k, n with $k \leq n$, there exists a k-selector with size $\mathcal{O}(k \log n)$.*

Proof. We use the probabilistic method. Fix some integer t and denote by S_i $(0 \leq i \leq t-1)$ a random set obtained by including each element of $[n]$ independently with probability $1/(k+1)$. Let $\mathcal{S} = \{S_0, \ldots, S_{t-1}\}$ be the family that we obtain.

Now, consider any two disjoint sets X, Y, where $|X| = x$, $|Y| = y$ and $k/2 \leq x \leq k, y \leq k$. Then, we can calculate the probability that S_i hits X and avoids Y.

$$\begin{aligned} Pr[S_i \text{ hits } X \text{ and avoids } Y] =& x \cdot \frac{1}{k+1}\left(1-\frac{1}{k+1}\right)^{x-1}\left(1-\frac{1}{k+1}\right)^{y} \\ =& \frac{x}{w}\left(1-\frac{1}{k+1}\right)^{x+y} \\ \geq& \frac{1}{2}\left(1-\frac{1}{k+1}\right)^{2(k+1)} \\ \geq& 1/32 \end{aligned} \tag{2.1}$$

The first inequality comes from the bounds on the size of X and Y, whereas the second occurs as the expression is minimized for $k = 1$. Thus, the probability that any set S_i of the family neither hits X nor avoids Y is $(31/32)^t$. In order to estimate the probability that the random family $\mathcal{S}$ is not a k-selector, we have to sum over all possible pair of sets X and Y.

$$Pr[\mathcal{S} \text{ is not a } k\text{-selector}] \leq \sum_{x=k/2}^{k}\binom{n}{x}\sum_{y=0}^{k}\binom{n}{y}(31/32)^t \tag{2.2}$$

After some simple calculations and by setting $t = (4/\log(32/31))k\log n + 1$, we get that $Pr[\mathcal{S} \text{ is not a } k\text{-selector}] < 1$ and thus, some $\mathcal{S}$ will be a k-selector. □

Now, let us turn our attention to strongly k-selective families. First, Dyachkov and Rykov [28] showed that there exist strongly k-selective families of size $\mathcal{O}(k^2 \log n)$. A much simpler proof of this upper bound is provided in [36], where the author uses extensively the notion of r-cover-free families.

As for the lower bound on strongly k-selective families, we have a lower bound of $\Omega(\min\{n, k^2 \log n/\log k\})$ on their size. The n factor comes from the simple observation that the family consisting of all singletons is a strongly k-selective family for any $k \leq n$. This lower bound was first approached by Dyachkov and Rykov [29], which proved a weaker form of the bound, namely an $\Omega(nk^2/\log k)$ lower bound. The bound was strengthened by Chaudhuri and Radhakrishnan in [9] to $\Omega(k^2 \ln n/\ln k)$ for $k \leq n^{1/3}$. However, the constant involved was pretty large (specifically 100) and was later improved to 16 by Clementi et al. [23].

We should also mention proofs of existence for the general (n, k, r)-selectors. De Bonis, Gasieniec and Vaccaro [6] proved that there exist (n, k, r)-selectors of size $\mathcal{O}(\frac{k^2}{k-r+1}\log(n/k))$. Notice that this bound means that there exist k-selectors of size $\mathcal{O}(k\log(n/k))$, which is proved to be optimal. In the same paper, the authors also showed a complicated lower bound of $\Omega(\min\{n, \frac{(r-1)^2}{k-r+1}\frac{\log(n/(k-r+1))}{\log((r-1)/(k-r+1))}\})$ on the size of (n, k, r)-selectors. This bound was simplified in [16] to form an $\Omega(\min\{n, \frac{k^2}{k-r+1}\frac{\log(n/k)}{\log(k/(k-r+1))}\})$ lower bound.

2.3 Explicit construction of selectors

The previous section referred to the probabilistic proof of existence of selectors with small size. However, this implies that the use of such a selector in a broadcasting or gossiping protocol leads to a non-constructive algorithm. Thus, in this section we will deal with the explicit construction of different kids of selectors.

After introducing the notion of k-selective families, Chlebus et al. [15] provided the construction of a $2^{\lceil m/6 \rceil}$-selective family of size $\mathcal{O}(2^{5m/6})$. For the construction of such a family, the authors used the notion of *deterministic sample*. Let $[n] = \{1, 2, \ldots, n\}$ and denote by $\mathcal{P}_r(n)$ the family of all subsets of $[n]$ with size r. Then,

Definition 2.3.1. *A **deterministic sample** is a member of the set*

$$DetSamples(r, m) = \{(D, \alpha) : \; D \in \mathcal{P}_r(m), \alpha \in \{0, 1\}^r\}$$

Definition 2.3.2. *Let $(D, \alpha) \in DetSamples(r, m)$. $STRINGS(D, \alpha)$ is defined as the set of all binary strings s such that $|s| = m$ and $\forall i,\, i \in D : \; s_i = \alpha_i$.*

Let us provide an example for better understanding of the deterministic samples. Assume that $m = 4$ and $r = 2$. Then, $[m] = \{1, 2, 3, 4\}$. Next, consider a deterministic sample of $DetSamples(2, 4)$, the pair $(\{1, 3\}, < 0100 >)$. It is clear that $A = STRINGS(\{1, 3\}, < 0110 >)$ consists of all strings of length 4 that have their first bit equal to 0 and the third equal to 1. Thus, only bits 2 and 4 can be chosen freely and as a result A consists of $2^2 = 4$ strings.

$$A = \{< 0010 >, < 0011 >, < 0110 >, < 0111 >\}$$

The crucial point about deterministic samples is that we can use them for the construction of k-selective families. Indeed,

Lemma 2.3.3. *The family $\mathcal{F} = \{STRINGS(D, \alpha) : (D, \alpha) \in DetSamples(r, m)$ is a 2^r-selective family for the set $\{0, 1\}^m$.*

Proof. Consider the set $X : \; |X| \leq 2^r$ and $r \geq 1$ with binary strings of length m. We will show how we can construct a deterministic sample $(D, \alpha) \in \mathcal{F}$ with the property $|STRINGS(D, \alpha) \cap X| = 1$.

Initially, let $X' = X$, $D = \emptyset$ and $\alpha = \epsilon$. We then seek the smallest index i such that two binary strings in X' differ in the ith bit (denote it with w). Consider now the sets $X_i = \{s \in \{0, 1\}^m : s_w = i\}$ for $i \in \{0, 1\}$. Clearly, $X_i \neq \emptyset$ and $X_0 \cap X_1 = \emptyset$. Let $k \in \{0, 1\}$ such that $|X_k| \leq |X_{1-k}|$. Considering that $|X_k| + |X_{1-k}| = |X'|$, we have that $0 < |X_k| \leq |X'|/2$. Next, let $X' = X_k$, $D = D \cup \{w\}$ and $\alpha = \alpha \cdot < k >$.

We repeat the above procedure until X' has just one element (if α is not long enough, we pad it to reach length m). Since in every repetition we halve the initial set, the number of repetitions needed is $\log 2^r = r$. By construction, $|STRINGS(D, \alpha) \cap X| = 1$. $\square$

The question now is what is the size of the selective family $\mathcal{F}$. By simple combinatorial arguments, it is easy to show that

Lemma 2.3.4. $|DetSamples(\lceil m/6 \rceil, m)| = \mathcal{O}(2^{5m/6})$.

By using now the standard mapping $\{0,1\}^m \to [2^m]$ and the above lemmas, we can construct a $2^{\lceil m/6 \rceil}$-selective family of size $\mathcal{O}(2^{5m/6})$, which we denote by $\mathcal{F}_m$.

De Marco and Pelc [62] managed to construct a smaller selective family. The authors used a different combinatorial tool, a Δ-disjunct family.

Definition 2.3.5. *A family $\mathcal{C}$ of subsets of $[n]$ is called* ***Δ-disjunct*** *if for every choice of $C_0, C_1, \ldots, C_\Delta \in \mathcal{C}$,*

$$C_0 \nsubseteq \bigcup_{i=1}^{\Delta} C_i.$$

Theorem 2.3.6. *For any integer $n > \Delta$, there exists a Δ-disjunct family of n subsets of $[t]$, where $t = \mathcal{O}(\Delta^2 \log n)$.*

The theorem guarantees the existence of Δ-disjunct families. The initial proof was based on the probabilistic method and thus there was no explicit way of constructing such a family. However, a way to construct this family was later found. Based on theorem 2.3.6 and the properties of Δ-disjunct families, we are now in position to construct a Δ-selective family.

Lemma 2.3.7. *For any positive integers $n > \Delta$, there exists a Δ-selective family of subsets of $[n]$ of size $\mathcal{O}(\Delta^2 \log n)$.*

A construction of even smaller k-selective families was provided by Clementi et al. in [21]. The authors proposed an algorithm that constructs a k-selective family of size $\mathcal{O}(\min\{n, k \log k \log(n/k)\})$. Clearly, this is only a $\log k$ factor away from the optimal size of k-selective families. The approach they used was to prove the existence of such a family probabilistically and then use derandomization.

Constructive algorithms have also been proposed for strongly k-selective families and k-selectors. Kautz and Singleton in [48], using superimposed codes, showed how to construct a (n, k, k)-selector of size $\mathcal{O}(\min\{n, k^2 \log^2 n\})$. Indyk [47] constructed k-selectors of size $\mathcal{O}(\min\{n, k \cdot polylog(n)\})$. The construction was based on a category of bipartite graphs called *dispersers*. Thus, we can construct k-selectors of size only a $polylog(n)$ factor larger than the optimal size. Finally, Chlebus and Kowalski [16] provided a construction for a generalized (n, k, r)-selector of size $\mathcal{O}(\min\{n, \frac{k^2}{k-r+1} \cdot polylog(n)\})$

Chapter 3

Broadcasting

Broadcasting is probably the most fundamental task in a network. The problem assumes that a specific node, called the *source*, holds some information which has to distributed to all the nodes of the network. We will refer to this piece of information as the *message* and we will denote it by m. Clearly, we are interested in communicating the message to nodes reachable from the source and thus we will assume that the graph has one connected component.

Broadcasting is the most well-studied problem in the setting of radio networks. The study of algorithms for broadcasting in radio networks has started from the seminal paper of Chlamtac and Kutten [12] and continues until today, as many questions remain open.

In this chapter, we will focus on three basic models of broadcasting: *centralized broadcasting, deterministic distributed broadcasting* and *deterministic randomized broadcasting.* For each model, we will first provide a brief description and then we will present broadcasting protocols, lower bounds, as well as examine some other variants. Finally, we will briefly present the research in *fault-tolerant* broadcasting and *geometric radio networks.*

3.1 Centralized Broadcasting

In this section we will consider broadcasting in radio networks with known topology, which means that the algorithms presented use complete information about the network topology. Such broadcasting algorithms are called *centralized* algorithms. This type of communication algorithms is useful in radio networks where the topology is reasonably stable. We also need the assumption that the topology of the network does not change during the execution of the algorithm.

Let us provide some more details on how a centralized broadcasting algorithm works. We assume that each node of the graph has a copy of the network graph and also knows its own label. The labels are distinct integers, which we assume to be consecutive naturals starting from 0 (the label of the source). We are interested in constructing a *broadcasting* *scheme* for the given network. A *broadcasting scheme* is a sequence of *transmission sets* (i.e. the set of nodes that transmit in each round), which, when applied to the network, guarantees that the message will reach all the nodes. Even though the scheme is executed in a distributed way, since each node has the same copy of the network, the scheme can be viewed as being designed centrally. Thus, we can consider the centralized algorithm as an algorithm with input the network graph and the source and output sets of nodes that transmit in each consecutive round (i.e. a broadcasting scheme).

The *length* of the broadcasting scheme is the number of rounds necessary for the scheme to inform every node of the network with the source message. The length may depend on two parameters, the size of the network n and the eccentricity D, which the largest distance from the source to a node in the graph. It is also important to distinguish between the running time of the algorithm that produces the broadcasting scheme and the length of the broadcasting scheme. We want to find an algorithm that runs in polynomial time and outputs the shortest scheme possible for a given network.

However, one may ask whether it is possible to find the optimal broadcasting scheme. Unfortunately, it has been proved in a seminal paper by Chlamtac and Kutten [12] that the problem of finding a shortest schedule for a given network is NP-hard. This hardness result, as it usually happens, has lead the research to follow two directions: to find *global bounds* on broadcasting time, which are bounds dependent only on parameters of the network, or to examine how well we can *approximate* of the optimal broadcasting time.

The first broadcasting algorithm that initiated the study of *global bounds* was presented in [10] and produced a broadcast schedule of length $\mathcal{O}(D\log^2(n/D))$. The next broadcasting algorithms tried to separate the D and $\log n$ terms in the upper bound. In [38], Gaber and Mansour presented a randomized scheme with expected length $\mathcal{O}(D + \log^5 n)$. The probabilistic algorithm can be slightly modified to produce a deterministic broadcast scheme of length $\mathcal{O}(D+\log^6 n)$. This result was further improved in [32], where the authors produced a randomized broadcast schedule with expected length $D+\mathcal{O}(\log^4 n)$ and a deterministic scheme with length $D+\mathcal{O}(\log^5 n)$. In [44], the authors provided an even better randomized algorithm which produced a scheme with expected length $D+\mathcal{O}(\log^2 n)$ and a corresponding deterministic scheme of length $D + \mathcal{O}(\log^3 n)$.

However, since there existed a lower bound of $D + \Omega(\log^2 n)$ on the length of any broadcasting scheme, a gap still remained on the time complexity of centralized broadcasting. While the gap has not yet been closed, further progress has been made. In [20], the authors presented a broadcasting algorithm with schedule length $D + \mathcal{O}(\frac{\log^3 n}{\log\log n})$, whereas in [52], Kowalski and Pelc came up with an algorithm that produces a scheme of length $\mathcal{O}(D+\log^2 n)$. A diagram presenting the development of algorithms for centralized broadcasting can be seen in figure 3.1.

As far as approximating the optimal broadcasting scheme is concerned, we have mostly negative results presented by Elkin and Kortsarz in [30] and later in [31].

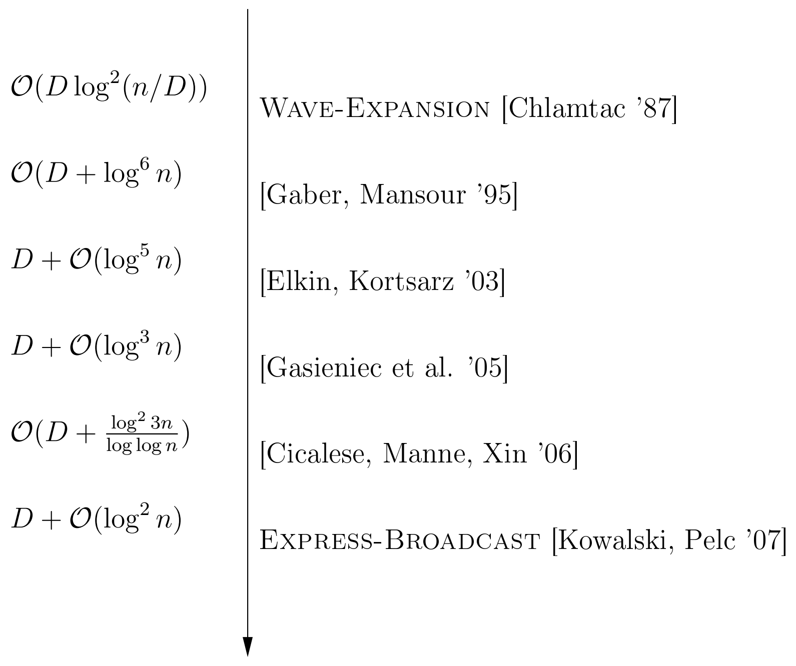

$\mathcal{O}(D \log^2(n/D))$
WAVE-EXPANSION [Chlamtac '87]
$\mathcal{O}(D + \log^6 n)$
[Gaber, Mansour '95]
$D + \mathcal{O}(\log^5 n)$
[Elkin, Kortsarz '03]
$D + \mathcal{O}(\log^3 n)$
[Gasieniec et al. '05]
$\mathcal{O}(D + \frac{\log^2 3n}{\log \log n})$
[Cicalese, Manne, Xin '06]
$D + \mathcal{O}(\log^2 n)$
EXPRESS-BROADCAST [Kowalski, Pelc '07]

Figure 3.1: The development of centralized broadcasting.

3.1.1 Finding the shortest broadcasting scheme is NP-hard

In [12], Chlamtac and Kutten showed that the problem of computing the shortest broadcasting scheme for an arbitrary graph is NP-hard. The authors managed to reduce the 3XC problem, which is NP-complete, to the decision problem of finding a broadcasting schedule with length less than a given number. Let us first provide a definition of the 3XC problem.

Definition 3.1.1. *(*EXACT-COVER*) Given a collection $\mathcal{C}$ of subsets of a set $\mathcal{U}$, find a subcollection $\mathcal{C}^* \subseteq \mathcal{C}$ such that each element in $\mathcal{U}$ is contained in exactly one subset of $\mathcal{C}^*$.*

The 3XC problem is the restriction of EXACT-COVER to instances where the subsets of $\mathcal{C}$ are only sets of three members. We note that 3XC is a decision problem, i.e. we have to answer whether there is some subcollection of $\mathcal{C}$ that satisfies the properties stated. We want to reduce 3XC to the following problem, which is clearly a more restricted version of the problem of finding the optimal scheme.

Definition 3.1.2. *(*2-MIN-BROADCAST*) Given an undirected network $G = (V, E)$ and a node $s \in V$ as the source, find whether there is a broadcasting scheme of length equal or less than 2.*

Let us now provide the detailed reduction from 3XC to 2-MIN-BROADCAST. Given an instance of 3XC, we construct the network $G = (V, E)$ in the following way. The set of nodes V includes the source s, a node c for each $c \in \mathcal{C}$ and a node u for every $u \in \mathcal{U}$. The set of edges consists of edges connecting the source to every node c and of edges connecting a node c with a node u when $u \in c$. A schematic example of the reduction is depicted in figure 3.3.

$$\mathcal{C} = \{a, b, c\}$$
$$\mathcal{U} = \{1, 2, 3, 4, 5\}$$
$$a = \{1, 2, 3\}$$
$$b = \{3, 4, 5\}$$
$$c = \{2, 4, 5\}$$

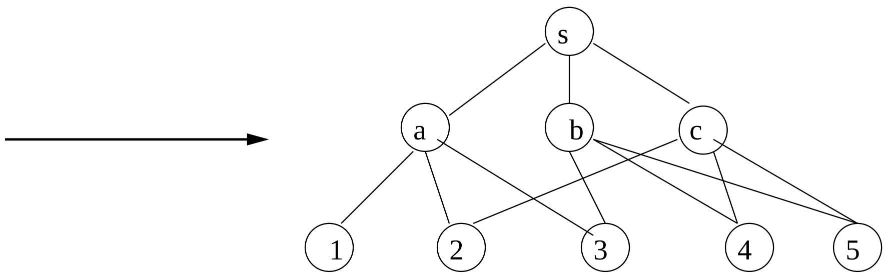


Figure 3.2: The reduction from an instance of 3xc to the corresponding instance of 2-Min-Broadcast.

Now, let us consider a solution of the 3xc problem. Then, the broadcasting can be performed in two rounds. In the first round, the source s transmits the message and every node $c \in \mathcal{C}$ receives the message. In the second round, only the nodes c participating in the optimal solution $\mathcal{C}^*$ transmit. As each node u is a member of exactly one subset of $\mathcal{C}^*$, the message will reach all the nodes $u \in \mathcal{U}$ and thus we have obtained a broadcasting schedule of length 2.

In the opposite direction, consider a solution to the 2-Min-Broadcast problem and denote by C_B the set of nodes of $\mathcal{C}$ that make a transmission. Clearly, since the message will have reached all the nodes $u \in \mathcal{U}$ by round 2, the union of the sets $c \in C_B$ covers all the elements of $\mathcal{C}$. Furthermore, every node $u \in \mathcal{U}$ is matched with exactly one set c, as in any other case a collision would occur in round 2 of the broadcasting (note that the first round is dedicated to the transmission of the source).

Consequently, there is a polynomial-time reduction from 3xc to 2-Min-Broadcast and so the problem 2-Min-Broadcast is NP-complete (it is easy to see that a solution can be verified efficiently). The following theorem is then obtained.

Theorem 3.1.3. *The problem of finding the optimal broadcasting schedule for an arbitrary radio network is NP-hard.*

An interesting point is that the problem remains NP-hard even when we ask for a broadcasting schedule that minimizes the average time, over all nodes, needed to complete broadcasting. Moreover, the theorem implies that we need to look for algorithms that compute schedules which approximate the optimal solution. We will discuss in the next section how well can we approximate the optimal schedule.

3.1.2 Lower bounds and inapproximability results for broadcasting

We first have to note that the eccentricity D is a trivial lower bound for the broadcasting time. Furthermore, Alon et al. proved in [1] that there exists a family of n-node networks of eccentricity 2 for which any broadcasting schedule requires $\Omega(log^2 n)$ rounds. This bound holds even when we assume that the nodes have full knowledge of the topology of the network or even when we consider randomized communication. The authors used the *probabilistic method* to prove the existence of such a family of networks, so the family has not yet been constructed explicitly. By combining those two bounds, we obtain a lower bound of $D + \Omega(\log^2 n)$ for the broadcasting time.

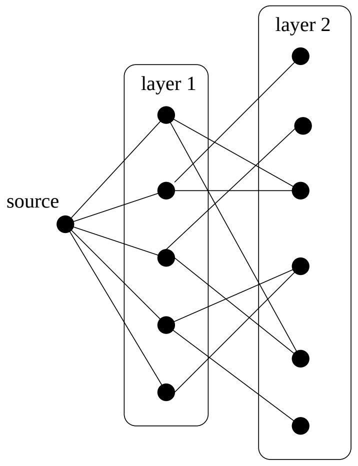


Figure 3.3: A graph of the family used for the $\Omega(\log^2 n)$ lower bound.

However, we should note a subtle difference about these two bounds. The factor D is a lower bound in the sense that it holds for every graph of eccentricity D. On the other hand, the $\Omega(\log^2 n)$ bound holds for a specific category of graphs with small eccentricity and not for every such graph. It is also important to note that the lower bound we have is weaker than a $\Omega(D + \log^2 n)$ bound, as the factor of D is equal to 1. That means that we are mostly interested in algorithms with complexity time that separates D from the $\log n$ factor. As we have already mentioned, most algorithms follow this motivation.

The concept of approximating the optimal broadcasting scheme for some graph G with length $opt(G)$ lays in a different direction of research. This time, we want to find an algorithm that produces an efficient scheme in relation with the best one and not by examining the universal bounds we have found. That means that the length of the approximating scheme will not depend on the parameters n and D, but on $opt(G)$. Nevertheless, approximating $opt(G)$ in a satisfactory way seems to be pretty difficult.

First, Elkin and Kortsarz [30] showed that approximating the optimal broadcasting time for an arbitrary n-node network by a multiplicative factor of $o(\log n)$ is impossible, unless $NP \subseteq BPTIME(n^{\mathcal{O}(\log \log n)})$. Later, the same authors proved in [31] that, under the same assumption, there exists a constant c such that there is no polynomial time algorithm which produces, for every n-node graph G, a broadcasting schedule of length at most $opt(G) + c \log^2 n$. Note here that the second inapproximability result has a $c \cdot \log^2 n$ additive approximation ratio, whereas in the first result the approximation factor of $o(\log n)$ is multiplicative. The existence of inapproximability results for both multiplicative and additive ratios shows the hardness of approximating the optimal broadcasting scheme within a satisfying factor.

3.1.3 A first approach to centralized broadcasting

The first to provide an algorithm for centralized broadcasting were Chlamtac and Weinstein [10]. The algorithm Wave-Expansion-Broadcast (WEB) they introduced runs in polynomial time and produces a broadcast scheme of length $\mathcal{O}(D \log^2(n/D))$. We use the term *wave expansion* to describe the idea behind the algorithm, as the progress of broadcasting can be viewed as a wave front carrying the message and advancing from the source node to the farthest node.

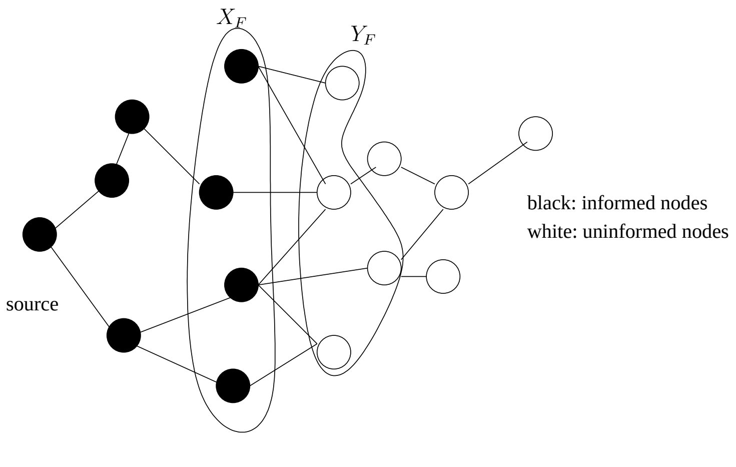


Figure 3.4: A wave front. X_F is the set of the potential transmitters, whereas Y_F is the set of the potential transmitters.

Let us describe the *wave front* in a more formal way. At each round of the algorithm, denote by X the set of informed nodes and by Y the set of the uninformed nodes. The *front* of this round is the set $F = \{(x, y) \ : x \in X, y \in Y, x \ neighbor \ of \ y\}$. We also define the set of *potential transmitters* $X_F = \{x \in X, \exists y \in Y \ : (x, y) \in F\}$ and the set of *potential receivers* $Y_F = \{y \in Y, \exists x \in X \ : (x, y) \in F\}$. Moreover, let us define the *spokesmen set* $S \subseteq X_F$ as the set of nodes from X_F that transmit in the next round. For any spokesmen set S, denote by $R_S \subseteq Y_F$ the set of nodes from Y_F that receive the message when the nodes from S transmit. The crucial step is to find a polynomial-time algorithm that constructs a set S at each round such that R_S is relatively large. This way, we ensure that at each round a large enough number of nodes from the next level receive the message and thus progress is made. The authors provided such an algorithm (Spokesmen-Election) and proved the following lemma.

Lemma 3.1.4. *For each round, the algorithm* Spokesmen-Election *computes in polynomial time a spokesmen set S such that $|R_S| > |Y_F|/\ln|X_F|$.*

The above lemma states that we can find a set S such that at least a fraction $1/\ln|X_F|$ of the nodes that can potentially receive the message at that round will actually be informed. Before we describe the algorithm WEB, we provide a useful definition.

Definition 3.1.5. *A **BFS level (layer)** consists of all the nodes with the same distance from the source. We denote the BFS level j by $L_j = \{v : dist(source, v) = j\}$.*

Algorithm 1: WEB

The algorithm consists of D phases, which we call *superwaves*. Each superwave executes a procedure CW, which consists of a number of rounds called *waves*. The ith superwave is responsible to transfer the message from the nodes of layer L_{i-1} to all the nodes of layer L_i. Thus, in the beginning of the superwave, $X_F = L_{i-1}$ and $Y_F = L_i$. In consecutive waves, we apply Spokesmen-Election to obtain a spokesmen set S, which we use for transmissions. We then remove from Y_F the nodes that received the message and apply the algorithm for the new front. The waves continue until Y_F is empty, after which the next superwave $i+1$ begins.

Since the algorithm Spokesmen-Election finds a set S such that $|R_S| > |Y_F|/\ln|X_F|$, we can see that the ith superwave has no more than $(\ln|L_{i-1}| \cdot \ln|L_i|)$ rounds. The first superwave needs only one round, and thus the total length of the scheme will be $1 + \sum_{i=2}^{D} \ln|L_{i-1}| \cdot \ln|L_i|$. The length will be maximized when the levels are of equal size. As a result, we conclude that algorithm WEB has running time $\mathcal{O}(D\log^2(n/D))$.

3.1.4 Broadcasting through clusterizing

We will here present the first important result for centralized broadcasting, which was presented by Gaber and Mansour [38]. The authors described an algorithm that produces a randomized broadcasting scheme of expected length $\mathcal{O}(D + \log^5 n)$. The basic idea behind the algorithm is that we want the message to reach fast the distant parts of the graph, after which we can easily inform the remaining nodes. The algorithm uses the idea of *network partition*: the graph is first partitioned into *groups* (or *super-levels*), where each group consist of consecutive *BFS levels* and each group is then partitioned into *clusters*, which we construct carefully so that they hold specific properties. The scheme then tries to pass the message as fast as possible from cluster to cluster and in parallel inform all the nodes within each cluster. A crucial point is to ensure that a large enough number of clusters can be active at the same time without causing collisions. Let us now provide some more details on the broadcasting algorithm.

We first distinguish the nodes according to their distance from the source. We can now divide the graph to groups, where group G_i is the set of nodes $G_i = \{v : v \in L_j, (i-1) \cdot D/x + 1 \leq j \leq i \cdot D/x\}$ (x is a parameter which is specified accordingly to optimize the running time). Clearly, each group consists of D/x consecutive BFS levels. We can now proceed in constructing the clusters for each group. The goal of the clusters is to receive the message from the lowest level of the group and quickly transmit it to the highest level. Thus, we should guarantee that each cluster has at least one node at the lowest level and that the union of the clusters can cover the whole group. Furthermore, we ask that the message can reach all the nodes of the cluster in a relatively small time, thus we require that the clusters have small diameter. Finally, we want to broadcast in parallel through clusters of the same color and thus accelerate the broadcasting. As a result, we ask that the clusters can be colored by as few as possible number of colors.

The construction of the clusters can be divided into two stages. In the first stage, we define sets for each group G_i, where each set corresponds to a node at the lowest level of the group. The sets are constructed in a way that guarantees that each node at the highest level has an ancestor at the lowest level, such that the shortest path between them belongs to the set. More formally, for each node u in the lowest level of group G_i we define the set $S_u^i = \{v : v \in L_j, dist(u,v) = j - ((i-1)D/x + 1), (i-1)D/x + 1 < j \leq i \cdot D/x\}$.

The sets defined are not enough to guarantee that the broadcasting can be performed in parallel efficiently, as we cannot calculate the degree of overlapping between the sets. Thus, we have to build clusters by merging the sets. We first have to define what we mean by *coloring* of the clusters.

Definition 3.1.6. *A* ***coloring*** *of the clusters $C_1, \dots, C_m$ is as assignment of a color to each cluster, such that there is no edge connecting clusters with the same color.*

We will omit here the details of the cluster construction and we will only state the basic theorem.

Theorem 3.1.7. *Given the sets $S_1^i, \dots, S_m^i$, we can construct clusters $C_1^i, \dots, C_k^i$ with the following properties:*

1. *Each cluster is the union of some sets.*
2. *Each set participates in at least one cluster.*
3. *The diameter of each cluster is $\mathcal{O}(D/x \cdot \log n)$.*
4. *There is a coloring of the clusters with $\mathcal{O}(\log n)$ colors.*

5. *The clusters can be constructed in polynomial time.*

The next step is to construct the *clusters' graph*, which is obtained by mapping each cluster to a node and introducing an edge between two nodes if in the graph there is some edge that connects nodes from the corresponding clusters or if they share a common node. On the clusters' graph, we then build a directed spanning tree of depth x, where each edge is directed from a cluster of lower level to a cluster of higher level. Each node assigns a high priority to one of its sons and a low priority to the others. The message moves quickly to nodes with high priority, thus we assign the priorities in such a way that every path form the source to some cluster contains few nodes with low priority. That way, the message reaches the most distant area of the graph in $\mathcal{O}(D)$ time.

More specifically, each cluster chooses a unique cluster from the preceding group from which the message will be received, which we call the *sender*. The procedure of picking a sender is applied to the clusters of each group, starting from the last group up to the first group G_1. Each cluster of group i knows all the clusters of the next level that have chosen it as a sender, as well as the *representatives* nodes chosen from the highest level of G_i. The procedure consists of the following stages:

1. The cluster C^i_j calculates its own *rank* according to the rank of its sons (the ranking procedure is described in subsection 3.1.5).

2. The cluster chooses a single node from the representatives as the *chosen representative* according to the ranking procedure.

3. The cluster chooses a node from the lowest level as the *messenger*. The messenger is essentially the node which receives the message from a cluster of the preceding group and forwards it directly to the chosen representative.

4. The cluster picks its *representative*, which is a node from the preceding group that propagates the message to the messenger.

A characteristic example of a clusters' graph is depicted in figure 3.6. We are now in position to describe the full broadcasting schedule. The broadcasting starts from the single cluster of the group G_1 which contains the source. The message is then forwarded from clusters in G_i to clusters in G_{i+1}, along the edges from the representative nodes of G_i to the messenger nodes of G_{i+1}. Within a cluster, the message is transmitted in two ways. We use a fast way to push quickly the message from the messenger to the chosen representative and a slow way to send the message to all the nodes of the cluster. We need a fast transmission within the cluster, so that we can guarantee that the message will reach all the messengers of the clusters in a relatively short time. This way, the message will be forwarded to the most distant parts of the graph, even when some nodes of lower levels are still uninformed.

These tasks are performed in an interleaved way, so that no collisions occur. More specifically, when the messenger receives the message from a cluster of a preceding group, each cluster initiates three parallel tasks:

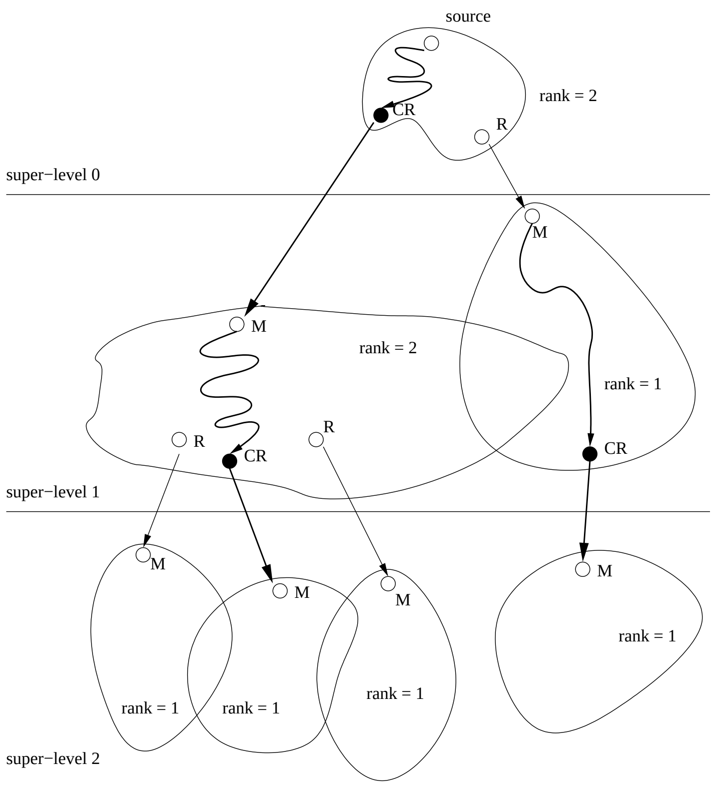


Figure 3.5: An example of a clusters' graph, along with the *messengers, representatives and chosen representatives*. We have also noted the ranks for each cluster.

Broadcast-Through: The message is passed along the path from the messenger to the chosen representative (the path is always within the cluster). Assume that the cluster has some color j. When the messenger gets the message, we wait until the first round $r \equiv 3j(\mod 3 \cdot \lceil 2 \log n \rceil)$. We then start to broadcast the message along the path, where each node transmits upon receiving the message.

Broadcast-All: The message is broadcasted to all the nodes of the cluster. In order to accomplish this, we use the randomized algorithm from [2], which has expected running time $\mathcal{O}(D \log n + \log^2 n)$. That means that there exists some schedule with execution time $K \cdot (D \log n + \log^2 n)$ for some constant K. Thus, we wait until some round $r \equiv 0(\mod K \cdot D/x \cdot \log^2 n)$ and we then apply the schedule. The schedule needs $K \cdot D/x \cdot \log^2 n$ rounds so that all nodes (the representatives included) are informed.

Group-to-Group: We wait until the message reaches a representative and then until the first round $r \equiv 0(\mod K\lceil \log^2 n \rceil)$. Then, all the representatives apply the randomized algorithm used for Broadcast-All. After $\mathcal{O}(\log^2 n)$ rounds, the messengers of the next group will have received the message.

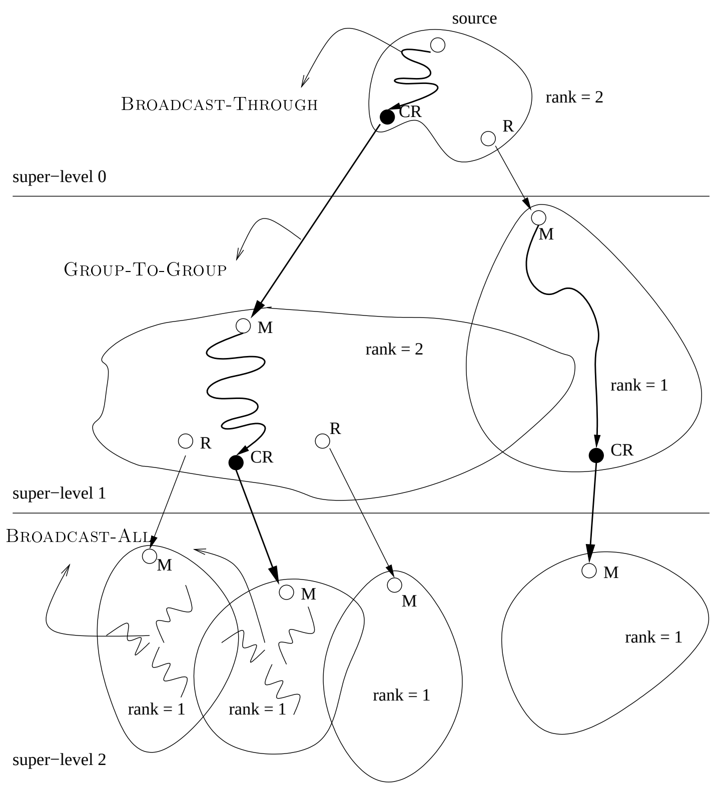


Figure 3.6: The application of the three interleaved tasks Broadcast-Through, Broadcast-All, Group-To-Group on a clusters' graph.

Our last concern is how to apply the three tasks in parallel without any collisions. The solution is to use time division and use a different clock for each task. Broadcast-through transmits only at rounds $r \equiv 0(\mod 3)$, Broadcast-All when $r \equiv 1(\mod 3)$ and Group-to-Group when $r \equiv 2(\mod 3)$. By interleaving the tasks in such a way, we ensure that each task completes without interrupting the other two tasks. The analysis of the algorithm leads to the following theorem.

Theorem 3.1.8. *There exists a randomized polynomial time algorithm that for any graph G computes a schedule of length $\mathcal{O}(D + \log^5 n)$.*

Elkin and Kortsarz in [32], based on the algorithm of the previous subsection, managed to construct a scheme that improved the broadcasting time to $\mathcal{O}(D+\log^4 n)$. Specifically, the algorithm completes broadcast in time $D + \mathcal{O}(\sqrt{D}\log^2 n)$. The idea is similar to the algorithm of Gaber and Mansour [38]. The authors partition the graph into *superlevels*, which are then further partitioned. However, the superlevels are not disjoint, as they may overlap. That consists an important difference between the two algorithm and allows the broadcasting to be performed more efficiently (the additive term is improved from $\mathcal{O}(\log^5 n)$ to $\mathcal{O}(\log^4 n)$). The procedures used for the broadcasting scheme are basically the same, but are not interleaved using time division. Instead, the algorithm is designed carefully such that, even though we use the same clock, no collision occurs between the different tasks. Although that does not change the time complexity, it manages to remove any positive constant from the factor D and thus obtain a complexity of the form $D + \mathcal{O}(polylog(n))$ (and not of $\mathcal{O}(D + polylog(n))$). For further information about the algorithm, [32] provides a detailed description of the partition and the broadcasting schedule.

3.1.5 Faster broadcasting

Gasieniec et al. [44] presented an efficient deterministic algorithm that computes a radio schedule of length $D + \mathcal{O}(\log^3 n)$. They also proposed a randomized algorithm that computes a schedule of length $D + \mathcal{O}(\log^2 n)$ with high probability. The algorithm is based on the notion of the *gathering spanning tree*. Thus, we need to describe how to construct such a spanning tree for an arbitrary graph. We will first need to introduce the simpler notion of the *rank* of a tree, as describe in [24].

Definition 3.1.9. *Consider an arbitrary tree T. We define that each leaf v has $rank(v) = 1$. Consider a node w which is not a leaf and its children $r_1, \ldots, r_m$. Define $r_{max} = \max_i r_i$. If w has a unique child with rank equal to r_{max}, then $rank(w) = r_{max}$. Else, $rank(w) = r_{max} + 1$.*

An important lemma about the rank of a tree is stated below.

Lemma 3.1.10. *The maximum rank of a node of a tree of size n is bounded by $\lceil \log n \rceil$.*

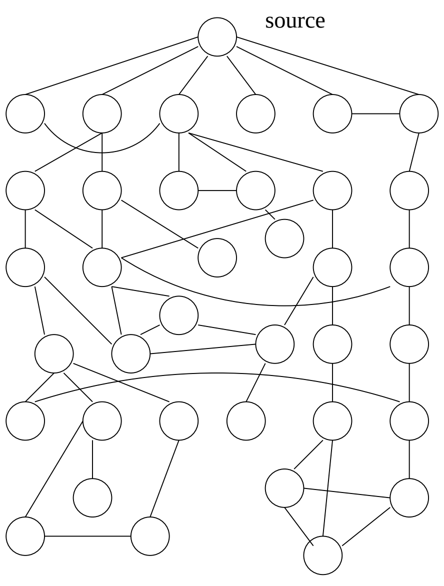
Definition 3.1.11. *We define the following sets:*

- *rank sets $R_i = \{v, rank(v) = i\}$ where $1 \leq i \leq \lceil \log n \rceil$.*
- *fast transmission sets $F_i^k = \{v, v \in l_k \cap R_i \, and \, parent(v) \in R_i\}$.*
- *slow transmission sets $S_i^k = \{v, v \in l_k \cap R_i \, and \, parent(v) \in R_j, j > i\}$.*

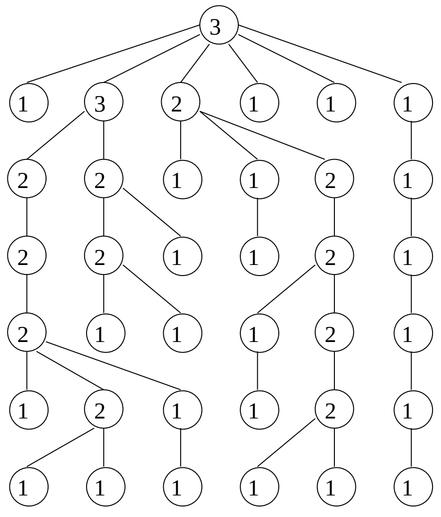
One would ask why the above definitions are useful for the broadcasting. The key point is that a fast transmission set holds an important property. During round i, all nodes in set F_i^k (for any $k = 1, \ldots, D$) can transmit simultaneously and the messages will reach their parents without any collisions. However, our goal is to adopt the above ideas for general graphs and thus the concept of a *gathering spanning tree (GST)* is introduced.

Definition 3.1.12. *In an arbitrary graph G, a GST is a BFS spanning tree T_G of G with the following properties:*

1. *The root of T_G is the source node.*
2. *T_G is ranked.*
3. *All nodes in F_i^k of T_G can transmit their message to their parents at the same round i without any collision and for any $1 \leq i\lceil \log n \rceil, 1 \leq k \leq D$.*



(a) The initial graph.



(b) The pre-gathering spanning tree.

Figure 3.7: The first stage of constructing a GST for an arbitrary graph.

Thus, a GST ensures that the key property of "fast transmission" holds even for general graphs. Furthermore, we can construct a GST for any graph G in polynomial time. The algorithm consists of two stages. In the first stage, we build an arbitrary ranked BFS spanning tree T_{PGT} of the graph rooted in the source, which is called a *pre-gathering-tree* (see figure 3.7). It is clear that T_{PGT} may violate the last property of a gathering spanning tree, as some transmissions of the fast transmission set may cause collisions. Consequently, we need to locate the pairs of nodes causing the collisions and then rearrange the edges between consecutive levels of T_{PGT}. For the detection of potential collisions, we use the procedure Check-Collision, which takes as a parameter a fast transmission set and returns a pair of node that causes some collision, else NULL. The detailed algorithm that produces a gathering spanning tree from a pre-gathering-tree is presented in figure 3.

An example of the construction of a GST from a pre-gathering spanning tree can be seen in figure 3.8. Once we have constructed the GST of the graph, we can describe the algorithm for the broadcasting. The algorithm uses the fast and slow transmission sets of the GST (see figure 3.9) However, we note that the construction of the GST deals with the case that the messages pass from the node to their parents, whereas in broadcasting

Procedure CHECK-COLLISION(i,j)

```
if $\exists u, v \in F_j^i$ and $(u, parent(v)) \in E$ then
    return $(u, v)$
else
    return NULL
end
```

Procedure GATHERING-SPANNING-TREE(T_{PGT})

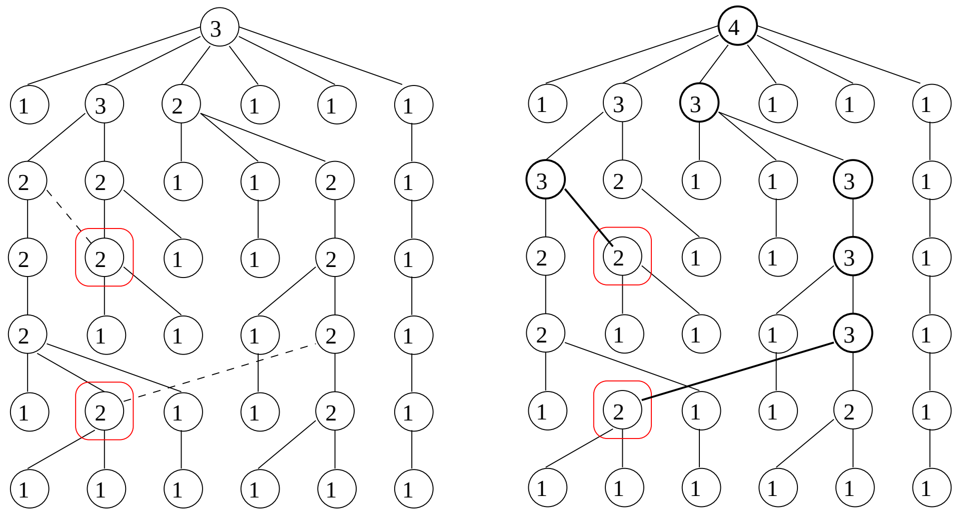
```
for $i := D$ downto 1 do
    for $j := r_{max}$ downto 1 do
        while CHECK-COLLISION(i,j) $\neq$ NULL do
            $rank(parent(v)) = j + 1$ ;
            remove the nodes $u, v$ from $F_j^i$ and add them to $S_j^i$ ;
            remove from $T_{PGT}$ the edge $(u, parent(u))$ ;
            add to $T_{PGT}$ the edge $(u, parent(v))$ ;
            update the ranks of the nodes at the top BFS layers ;
            recompute the fast and slow transmission sets ;
        end
    end
end
```

we want the message to propagate in the opposite direction. Considering that the graph is undirected, we can reverse the direction of the transmissions within the same stage (i.e. the transmissions between nodes of the same rank go from $parent(v)$ to v) and thus guarantee that no collisions happen during a fast transmission. The broadcasting schedule uses a different mechanism for the slow and the fast transmissions.

Fast Transmissions: Consider a node v of rank j on BFS layer l_i which has a child w of the same rank at the next BFS layer. Then, v transmits in round $r \equiv i + 6j(\mod 6r_{max})$.

Slow Transmissions: The message has to be transmitted from node u to some node of lower rank at the next BFS layer. We use then the transmission procedure CW proposed by Chlamtac [10]. This procedure can move the message from one partition of a bipartite graph (BFS layer l_i) to the other partition (the next level l_{i+1}) in time $\mathcal{O}(\log^2 n)$. We run the procedure periodically at every BFS layer of the GST, but we perform transmissions for layer i only at rounds $r \equiv i + 3(\mod 6)$.

Let us now proceed to the analysis of the algorithm. First, we note that the time division between the fast and slow transmissions ensures that no collisions occur between nodes from different BFS layers (two layers transmitting simultaneously have distance at



(a) The pre-gathering spanning tree. The nodes in red belong to the fast transmission sets, but collisions may occur. The dashed lines show the edges that lead to the collisions.

(b) The final gathering spanning tree after modifying edges and rankings.

Figure 3.8: The second stage of constructing a GST for an arbitrary graph.

least 3) and that the two procedures do not interact. Now, consider an arbitrary leaf a of the GST and the unique shortest path $p(a)$ from the source to the leaf. The message may not follow this path to reach the leaf a, but we focus our attention to the progress of the message along this path. We can think of the path $p(a)$ as a path consisting of segments $p_i^F(a)$ of fast transmission edges (edges where the end nodes are of the same rank) and of segments $p_i^S(a)$, where each $p_i^S(a)$ is an edge (u, w), u is at some BFS layer l_k, w is in l_{k+1} and $rank(u) > rank(w)$. Such edges will be called slow transmission edges. Now, we can view the traversing of the path as alternating between segments of fast transmission and slow transmission edges, thus:

$$p(a) =< p_1^F(a), p_1^S(a), \dots p_m^F(a), p_m^S(a) >$$

Note that a fast transmission segment can be null. When the message reaches the first node of a fast segment, it may have to wait at most $6r_{max} = \mathcal{O}(\log n)$ rounds before transmitting, but then the message propagates through the fast segment without any further delays. On the other hand, when the message has reached a node u such that the next edge (u, w) is a slow transmission edge, we may have to wait until the next execution of the procedure CW for that layer. Even in this case, the next execution guarantees that the message will move forward and consequently, the time needed will be at most $\mathcal{O}(\log^2 n)$.

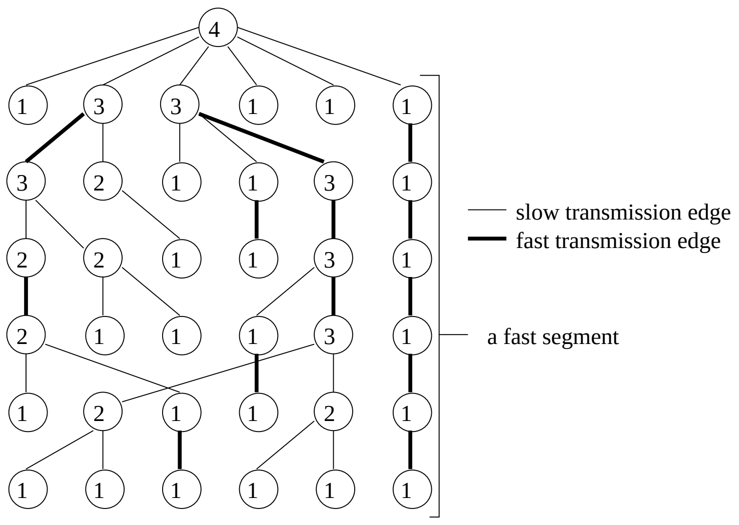


Figure 3.9: The figure shows the distinction between fast and slow transmission edges in a gathering spanning tree.

Let us sum up the time required for the message to traverse the path. Denote by D_i the length of the fast segment p_i^F. Then, for the fast transmissions we need time $\mathcal{O}(\log n) + D_1 + \ldots + \mathcal{O}(\log n) + D_{r_{max}} \leq D + \mathcal{O}(\log^2 n)$. The slow transmissions need time $r_{max} \cdot \mathcal{O}(\log^2 n) = \mathcal{O}(\log^3 n)$. As a result, the total length of the broadcasting is bounded by $D + \mathcal{O}(\log^3 n)$.

Theorem 3.1.13. *There exists a polynomial time algorithm that constructs a broadcasting schedule of length $D + \mathcal{O}(\log^3 n)$ for any arbitrary graph of size n and eccentricity D.*

We can now proceed one step ahead and replace the deterministic procedure CW with the randomized procedure RCW. Procedure RCW works for $\lceil \log n \rceil$ rounds for each BFS layer and is repeated in a periodic manner like CW. When RCW begins, only the informed nodes of the layer participate. In round $i, 1 \leq i \leq \lceil \log n \rceil$, each node participating transmits randomly and uniformly with probability $1/2^i$. It can be proved that RCW guarantees that the message reaches every node of the graph with high probability. Thus, we obtain the following theorem.

Theorem 3.1.14. *There exists a randomized algorithm that for any network of size n and eccentricity D completes broadcasting with high probability in time $D + \mathcal{O}(\log^2 n)$.*

3.1.6 Recent results in centralized broadcasting

Recently, Kowalski and Pelc [52] presented a deterministic polynomial-time algorithm that produces a broadcasting scheme of length $\mathcal{O}(D + \log^2 n)$ for any undirected graph of size n and eccentricity D. In view of the inapproximability results by Elkin and Kortsatz, the length is asymptotically optimal. We will not present the algorithm with full details, but we will only describe the main outline and some of the basic ideas involved.

The authors first provide an algorithm (FAST-DETERMINISTIC-BROADCAST) that produces a broadcast scheme of length $\mathcal{O}(D\log n + \log^2 n)$. Next, they combine FAST-DETERMINISTIC-BROADCAST along with the notion of the gathering spanning tree (GST) to construct an algorithm (EXPRESS-BROADCAST) that produces a scheme this time with length $\mathcal{O}(D + \log^2 n)$.

The algorithm FAST-DETERMINISTIC-BROADCAST is divided into *phases*. For phase k, we define a partition $S_{k,j}$ ($j \leq D$) of the set $\bigcup_{j>1} L_j$. The set $S_{k,j}$ contains all the nodes u, such that the last informed node on any shortest path from the source to u belongs to level L_j. Furthermore, we denote by $R_{k,j}$ the set of the informed nodes of L_j which are witnesses for the nodes in $S_{k,j}$. Phase k is responsible for moving nodes from set $S_{k,j}$ to set $S_{k+1,j+1}$. Equivalently, for a node $v \in S_{k,j}$, we must inform at least one of its predecessors in level L_{j+1}. Since it would be time consuming to move all the nodes from $S_{k,j}$, our goal is to move only a fraction of them. Specifically, we repeatedly use algorithm SPOKESMEN-ELECTION from [10] with input the bipartite graph constructed from sets $R_{k,j}$ and $S_{k,j}$. The spokesmen set produced is then used for transmissions. The algorithm is repeated $2\log n$ times, so that we can ensure that at least half nodes will move to set $S_{k+1,j+1}$.

It can be proved that $\mathcal{O}(D+\log n)$ phases are enough to complete broadcasting. Since each phase lasts $\mathcal{O}(\log n)$ rounds, the running time of FAST-DETERMINISTIC-BROADCAST (FDB) is $\mathcal{O}(D\log n + \log^2 n)$. In order to further decrease the length of the scheme, we have to use FAST-DETERMINISTIC-BROADCAST in parallel with a gathering spanning tree. Through the slow transmission edges we use algorithm FDB, whereas through a fast transmission segment the message can be forwarded fast without any collisions.

Specifically, the authors managed to construct a graph G' from G, where the fast transmission segments of the GST correspond to single nodes in G'. Graph G' now has diameter at most $\mathcal{O}(D/\log n + \log n)$ and thus algorithm FDB produces a scheme of length $\mathcal{O}(D + \log^2 n)$. Now, the produced scheme has to be modified to get a scheme for graph G, but since the fast transmission edges are easily dealt with, the length of the final scheme does not increase. Thus, we obtain the following theorem.

Theorem 3.1.15. *Algorithm* EXPRESS-BROADCAST *runs in polynomial time and produces a broadcasting scheme of length* $\mathcal{O}(D + \log^2 n)$.

As we have already mentioned, EXPRESS-BROADCAST is asympotically optimal. However, the schemes for the previous sections are of the form $D + polylog(n)$, i.e. the constant factor for D is exactly 1. Thus, the question whether we can obtain a scheme of length $D + \mathcal{O}(\log^2 n)$ remains open. Furthermore, it would be interesting to find an algorithm that produces a scheme of length $\mathcal{O}(opt(G)\log n)$.

Finally, in [20], the authors provided an algorithm that outputs a schedule of $D + \mathcal{O}(\frac{\log^3 n}{\log\log n})$ time. For large enough D, the schedule performs even better than the schedule of [52]. The authors based their algorithm on the notions of *tree ranking* and the *gathering spanning tree*. However, they proposed the stronger notion of *super-ranking* and respectively the *super-gathering spanning tree (SGST)*, which extend the previously defined notions and hold more useful properties. The SGST can be constructed in polynomial time from a simple GST and then can be used in order to perform broadcasting more efficiently.

3.2 Distributed Deterministic Broadcasting

Assuming that the topology of a network is known to the nodes is not always realistic. The decentralized nature of modern networks, in combination with the fact that a network might not always be stable or could be dynamically changing, can make impossible to implement a centralized scheduling scheme. As a result, we need a model where every node holds as little information as possible about the network.

We will study the model where, at the initial situation, the knowledge of each node is limited to its own unique identifier. The nodes are not aware of their neighbors and, as a result, every distributed algorithm is not able to exploit these information. One could also argue that the distinct label each node holds are unnecessary, however, deterministic broadcasting is impossible to be completed if the radio network is anonymous (randomized broadcasting, on the other hand, is feasible). However, we will often provide the nodes with information such as the size of the network n or the eccentricity D.

Furthermore, we should make an important observation on how we measure the execution time of a broadcasting protocol in the setting of ad-hoc radio networks. We assume that the processing time of each node is insignificant compared to the time needed for the transmission of a message (equivalently, the duration of a time-slot). Thus, the execution or running time of a distributed protocol is actually the number of rounds necessary for the protocol to complete its task. We should compare this with the running time of a centralized algorithm, which referred to the processing time the nodes needed.

Before we provide a brief description of the research in deterministic distributed broadcasting, we should note an important variant of the communication model we consider. The standard model assumes that a node starts its transmission schedule after it receives the source message for the first time. Namely, the clock of a node (apart from the source) starts in the round when the node first receives the source message. We call this the *conditional wake-up* model. Alternatively, one may consider the *spontaneous wake-up* model, where all nodes are assumed to start their clocks when the source transmits for the first time. This means that the nodes may start transmitting before receiving the source message, which could be used for pre-processing and gathering of information. We will also refer to these two models as broadcasting without and with spontaneous transmissions respectively.

The first to deal with deterministic broadcasting in the setting of ad hoc networks was Chlamtac and Farago in [11]. The authors constructed an algorithm that completes broadcasting in time $\mathcal{O}(D\frac{\Delta^2}{\log^2\Delta}\log^2 n)$. Later, an algorithm working in time $\mathcal{O}(D\Delta\log^{\log\Delta} n)$ was constructed in [4]. Another interesting approach to the problem was by Clementi et al. [23], which provided a protocol with running time $\mathcal{O}(D\Delta\log^a(n/\Delta))$ for $a > 2$ (if the nodes know n, a decreases to 2 and further to 1 if Δ is also known). However, the above algorithms perform well only when Δ is relatively small, as otherwise the execution time is more than quadratic to n.

The first subquadratic algorithm was presented in [13] and completed broadcasting in $\mathcal{O}(n^{11/6})$ time. In the same paper, the authors presented a simple quadratic algorithm. After that publication, the bound was further improved in a very short time. First, De Marco and Pelc [62] improved the execution time and showed how broadcasting can be achieved in time $\mathcal{O}(n^{5/3}(\log n)^{1/3})$. Peleg also provided an unpublished algorithm with running complexity $\mathcal{O}(n^{3/2}\sqrt{\log n})$. In [15], Chlebus et al. proposed an even faster broadcasting algorithm with execution time $\mathcal{O}(n^{3/2})$. Next, a considerably faster algorithm was proposed in [18], completing broadcasting in time $\mathcal{O}(n\log^2 n)$.

In 2003, Kowalski and Pelc [54] presented a broadcasting algorithm with execution time $\mathcal{O}(n\log n\log D)$, where D is the eccentricity of the radio network. This bound is clearly an improvement over the bound obtained by Gaseniec et. al in [18], as the eccentricity is at the worst case $\mathcal{O}(n)$. The execution time was even further improved in [25] to $\mathcal{O}(n\log^2 D)$. However, the two last algorithms do not improve the worst case running time when $D = \Omega(n)$. Very recently, in 2008, De Marco [61] constructed a broadcasting algorithm with execution time $\mathcal{O}(n\log n\log\log n)$, which is the best deterministic distributed algorithm we have so far. We have to note here that all algorithms with broadcasting times less than $\mathcal{O}(n\log^2 n)$ are non-constructive, as they all use families of sets whose existence has been proven, but no efficient algorithm has been presented to produce them. Indyk [47] gave a constructive algorithm with running time $\mathcal{O}(n\log^{\mathcal{O}(1)} n)$. An interesting review on broadcasting in radio networks was recently presented by Peleg [67]. We can also see a schematic representation of the development of distributed broadcasting algorithms for radio networks in figure 3.10.

As far as the lower bound is concerned, first Bruschi and Del Pinto [8] proved a bound of $\Omega(D\log n)$. The bound was further improved to $\Omega(n\log D)$ by Clementi et al. [23]. These two lower bounds are for networks for arbitrary diameter D. If we restrict to networks with constant diameter, first Bar-Yehuda, Goldreich and Itai [2] claimed a linear lower bound on the broadcasting time. Nevertheless, the bound was proved to be incorrect (in fact it holds for a more pessimistic model of radio networks) by Kowalski and Pelc [55]. In the same paper, the authors proved a lower bound $\Omega(\sqrt[4]{n})$ on graphs with diameter 4, which was improved to $\Omega(\sqrt{n})$ in [7].

3.2.1 Broadcasting with acknowledgement

An important thing we should first consider is *acknowledged radio broadcasting (ARB)*. In simple radio broadcasting, the goal is to disseminate the message from the source to all the nodes of the network. In ARB, however, the goal is not only to achieve broadcasting,

$\mathcal{O}(n^2)$	SIMPLE-SEQUENCING [Chlebus et al. '00]
$\mathcal{O}(n^{11/6})$	SELECTIVE-BROADCASTING [Chlebus et al. '00]
$\mathcal{O}(n^{5/3}(\log n)^{1/3})$	[De Marco, Pelc '01]
$\mathcal{O}(n^{3/2}\sqrt{\log n})$	[Peleg, manuscript]
$\mathcal{O}(n^{3/2})$	SINGLE-PRIME [Chlebus et al. '00]
$\mathcal{O}(n \log^2 n)$	SELECTOR-BROADCAST [Chrobak, Gasieniec, Rytter '00]
$\mathcal{O}(n \log n \log D)$	[Kowalski, Pelc '03]
$\mathcal{O}(n \log^2 D)$	OBLIVIOUS-BROADCAST [Czumaj, Rytter '03]
$\mathcal{O}(n \log n \log\log n)$	[De Marco '08]

Figure 3.10: Development of broadcasting algorithms for the deterministic distributed setting.

but also to ensure that the source is informed about it. The motivation behind ARB is that we may need to know exactly when broadcasting ends, in order to be able to perform some other task afterwards.

Let us provide a more formal definition of acknowledged broadcasting. We say that an algorithm accomplishes ARB in t rounds, if all the nodes know the message and after round t, no more messages are transmitted and the source knows that all the nodes know the message. We should note here that when the source is informed about ARB, it easy to execute one more broadcast in order to let all the nodes know that the broadcasting is complete.

Clearly, ARB is a more difficult task than radio broadcasting. In fact, Gasieniec et al. [13] proved that ARB is *infeasible* in the standard distributed model without collision detection, even when we consider symmetric graphs. The model they examined assumes that each node knows only its own label and they even allowed spontaneous transmissions. On the other hand, assuming collision detection, the authors provided a broadcasting algorithm with time complexity $\mathcal{O}(nD)$, for arbitrary strongly connected networks.

Nevertheless, it was proved that the impossibility of acknowledged broadcasting is caused by the singleton network (containing only the source), as any algorithm is not capable of distinguishing that the source is alone. Specifically, Uchida, Chen and Wada [72] showed that if $n \geq 2$, we can construct algorithms that perform ARB in linear time for symmetric networks and in $\mathcal{O}(n^{4/3} \log^{10/3} n)$ rounds for strongly connected graphs.

The above results were further improved by Fusco and Pelc [37]. The authors presented an algorithm that performs ARB in the model with collision detection, working in time $\mathcal{O}(\min\{n \log^2 D, n \log n \log\log n\})$ for strongly connected networks. In the model without collision detection, they constructed a similar algorithm working in time $\mathcal{O}(n \log n \log\log n)$ for strongly connected networks of size greater than 2.

3.2.2 Lower bounds

Before we turn our attention to algorithms for distributed deterministic broadcasting, we will first present some lower bounds on broadcasting obtained so far. First, we will examine the $\Omega(D \log n)$ lower bound proved by Bruschi and Del Pinto [8]. The model the authors assume is the standard distributed model, where the nodes know only their label and also the parameters n and D. They also assume that we have no spontaneous transmissions. Moreover, the bound holds even when we consider undirected graphs.

We will construct a class of radio networks $\mathcal{C}$ and prove the lower bound on networks that belong to the class. A member C_n^D $(D \leq n/2)$ of class $\mathcal{C}$ is a layered network with $D+1$ layers. The first layer (layer 0) contains only the source node. Then, each of the layers $1, 2, \ldots, D-1$ contains exactly $h = \lfloor n/D \rfloor$ nodes. The last layer D contains the remaining nodes. Now, let us describe the edges between the nodes. The source is connected with all nodes of layer 1. All the nodes of layer i $(2 \leq i \leq D)$ are connected to two distinct nodes $< p_{i-1}, q_{i-1} >$ of layer $i-1$. We should also note that there are no edges between nodes of the same layer. An example of such a graph is depicted in figure 3.11.

Our goal is to prove that for an arbitrary protocol Π and parameters n and $D \leq n/2$, there exists a network C_n^D where Π needs $\Omega(D \log n)$ rounds to complete broadcasting. First, we should note that the absence of spontaneous transmissions and the topology of the network imply that broadcasting performs in a layer-to-layer way. That means that once a node of layer i first receives a message, all the other nodes of layer i also receive the message at that round for the first time. Thus, the total amount of rounds needed to complete broadcast is the sum of the number of rounds needed to transfer a message from layer i to layer $i+1$. We will first deal with the forwarding of a message from one layer to the next and show a lower bound on that.

The first step is to show a basic lemma.

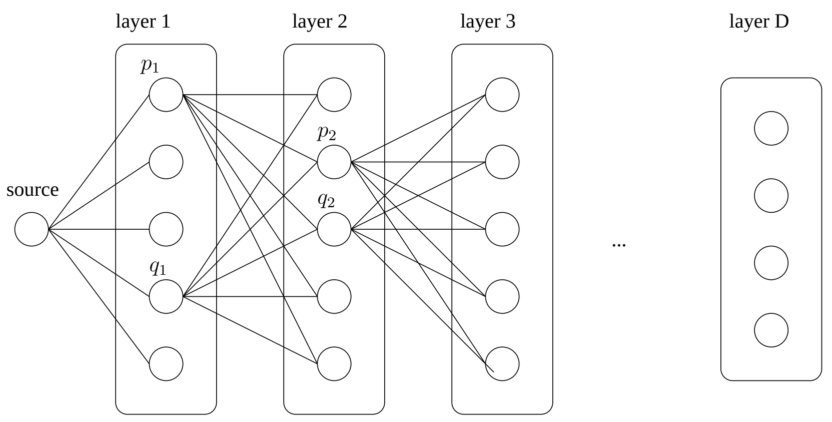


Figure 3.11: The graph C_n^D.

Lemma 3.2.1. *Let $S \subseteq \{1, 2, \ldots, n\}$ with $|S| = N \geq 2$ and a transmission schedule $T = \{T_1, \ldots, T_t\}$, where $t \leq \log(N/2)$. Then, we can find two elements $x, y \in S$ such that: $\forall j, 1 \leq j \leq t, |\{x, y\} \cap T_j| \neq 1$.*

The idea behind the proof of the lemma is the following. Consider the first transmission set T_1. Notice that $S \cap T_1$ is the set of nodes of S that transmit, whereas $S \cap \bar{T_1}$ is the nodes of S that do not transmit. Clearly, $|S \cap T_1| + |S \cap \bar{T_1}| = N$ and thus the largest of those sets (let it be S^1) is larger or equal to $N/2$. Furthermore, as long as $|S^1| \geq 2$, that means that S^1 contains two elements u, v such that neither of them or both of them belong to T_1. We have thus constructed a set that holds the desired property and is at least half the size of N. We can continue the process with the next transmission sets, each time halving the set to $S^2, S^3, \ldots$. Clearly, since we have at most $\log(N/2)$ transmission sets, the last set that occurs has size at least 2 and thus we can get two elements that satisfy the property.

Now, we can use the lemma to prove a lower bound on the number of rounds necessary to forward a message between two consecutive layers. Consider the last layer of C_n^D that have received the message and let S_{i-1} be the set of the labels of the nodes in layers $0, \ldots, i-1$. We will make the stronger assumption that each node receives, along with the message, the whole history of the network. Thus, each node knows exactly S_{i-1}. We will show that it is always possible to find an assignment of labels ϕ for p_i, q_i such that $\phi(p_i), \phi(q_i) \in S = \{\{1, \ldots, n\} - S_{i-1}\}$ and more than $\log((n-(i-1)h)/2)$ rounds are required to forward a message to layer $i+1$.

Notice that the protocol Π must guess the labels of p_i, q_i from the set S. We also know that $|S| = n - (i-i)h$. By using the above lemma, for any schedule T of length $t \leq \log((n-(i-1)h)/2)$, we can always find a pair of labels x, y such that no transmission is successful for that schedule. Let $\phi(p_i) = x$ and $\phi(q_i) = y$ be the assignment we need. Thus, by using this assignment, more than $\log((n-(i-1)h)/2)$ rounds are needed to forward the message to the next layer.

The next step is to extend the lower bound from two consecutive layers to all the layers of C_n^D. We can repeatedly use the assignment above to delay the forwarding of the message $\log((n-(i-1)h)/2)$ in layer i. Thus, we obtain a lower bound t, where

$$t > \sum_{i=1}^{D-1} \log\left(\frac{n-(i-1)h}{2}\right) = \Omega(n \log D)$$

after some calculations. We have thus proved the lower bound. We should note that the the general idea of the proof is described and some details are omitted.

Another lower bound for undirected networks is a $\Omega(n\frac{\log n}{\log(n/D)})$ bound presented in [53] by Kowalski and Pelc. The authors assume some given broadcasting algorithm and then construct a network on which it works slowly, by combining families of sets used for jamming potential messages and the notion of selective families. The network constructed from this procedure is not the typical complete layered network frequently used to prove lower bounds.

As far as broadcasting in *directed* graphs is concerned, the $\Omega(n\frac{\log n}{\log(n/D)})$ bound still holds. However, Clementi et al. [23] provided another lower bound of $\Omega(n \log D)$. It is also proved in [53] that the bound does not hold for the undirected version of the class of graphs considered, as we can construct a broadcasting algorithm which works in time $\mathcal{O}(n + D\log n)$ for all undirected complete layered n-node networks. The theorem proved in [23] is the following.

Theorem 3.2.2. *For any distributed protocol Π and any value of n and $D \leq n/6$, there exists a graph G^Π with size n and maximum eccentricity D, such that Π performs broadcasting in $\Omega(n \log D)$ rounds. The lower bound also holds when the nodes know n.*

Proof. The graph G^Π is a layered n-node graph with $D+1$ layers. The first layer (layer 0) contains only the source node. The layers $1, \ldots, D-1$ contain each no more than $\lfloor n/2D \rfloor$ nodes and the last layer D contains all the remaining nodes. Edges exist from all nodes of layer i to all nodes of layer $i+1$. Schematically, the graph is represented in figure 3.12.

Our goal is to find an assignment of labels for all nodes in such a way that any protocol Π requires $\Omega((n/D)\log D)$ rounds to transfer a message between two consecutive layers. In order to perform this assignment, we use *selective families*. Specifically, we use the lemma which states that, if $2 \leq D \leq n/6$, any $(\lceil n/2 \rceil, \lfloor n/2D \rfloor)$-selective family has size at least $T = \lfloor c(n/D)\log D \rfloor$, where c is some constant. This will help us prove that it is possible to find a label assignment to layers $0, \ldots, j$ such that Π does not reach layer j before the round with number $j \cdot T$. Thus, the last layer will not be informed before round $cD \cdot (n/D)\log D = c \cdot n \log D$.

We will show this claim inductively. Clearly, it holds trivially for $j = 0$. Assume that we have found a satisfying assignment for layers up to j and we want to find the assignment for layer $j+1$. Let R be the set of labels not already assigned to layers $0, \ldots, j$. Notice

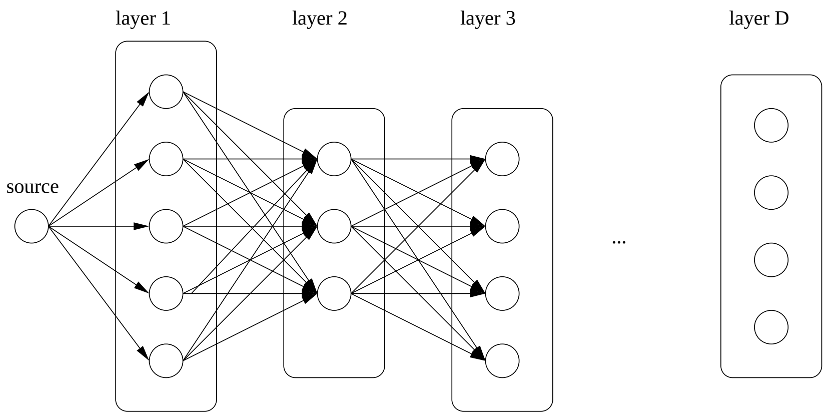


Figure 3.12: The graph G^{Π} with size n and eccentricity D.

that $|R| \geq \lceil n/2 \rceil$. Also, note that the behavior of the protocol Π is independent of whether we assign to layer $j+1$ the whole R or some subset L. Thus, define

$$F_t = \{v \in R | v \text{ transmits in round } j \cdot T + t\}$$

Now, we can construct the family $\mathcal{F} = \{F_1, \ldots, F_{T-1}\}$. Since $|\mathcal{F}| < T$, $\mathcal{F}$ is not a $(\lceil n/2 \rceil, \lfloor n/2D \rfloor)$-selective family. Consequently, there exists some set $L \subset R$ such that $|L| \leq \lfloor n/2D \rfloor$ and L is not *hit* by any set of $\mathcal{F}$ in any round $t : jT+1 \leq t \leq (j+1)T-1$. Choosing the set L as the assignment for layer $j+1$, the claim is proved. □

Now, let us turn our attention to broadcasting in graphs with constant eccentricity. We assume that the nodes know not only their labels, but also the labels of their neighbors (thus having a form of local knowledge). In [2], Bar-Yehuda et al. constructed a family of graphs with size n and diameter 3 and claimed that any broadcasting algorithm requires $\Omega(n)$ rounds to complete the task on one of these graphs. Moreover, the claim was further strengthened in [46], where a lower bound of $(n-1)$ rounds was proved for the same family of graphs as in [2]. However, the claim was proved to be wrong in [55] by Kowalski and Pelc, where the authors showed how to broadcast in logarithmic time in all the graphs from [2]. The linear lower bound actually holds for a more pessimistic model of communication in radio networks.

The subtle difference between the two models lays in the way we handle *collision detection.* The standard model (where the claim was proved to be incorrect) assumes that a node cannot distinguish a round with no transmission from a round where a collision occurs, whereas a collision is always distinguishable from a successful transmission. The more pessimistic model assumes that when a collision occurs at node u, the effect can be either the same as if no neighbor of u transmitted or the same as if any single neighbor of u transmitted, the choice of which is left to the adversary. This model has also a very natural interpretation: in the case of a collision, noise indistinguishable from the background noise may be produced or some of the messages transmitted may prevail and reach its target. As it seems, the standard model is a *weaker* model, since the nodes may be able to exploit the information of the absence of incoming messages.

Kowalski and Pelc in the same paper ([55]) and under the same assumption constructed a class of graphs of eccentricity 4, such that every broadcasting algorithm requires $\Omega(n^{1/4})$ rounds on one of these graphs. The lower bound was further improved by Brito et al. [7] to $\Omega(\sqrt{n})$. The authors showed that for every deterministic broadcasting protocol, there exists a network with $D = 2$ which requires at least $\Omega(\sqrt{n})$ rounds to complete broadcasting. For networks with arbitrary eccentricity, the lower bound can be extended to $\Omega(\sqrt{nD})$ rounds. The approach followed by the authors lays in a different direction compared to previous approaches. The key idea is to quantify the amount of connectivity information concerning the topology of the network that the source can learn in a number of rounds. By obtaining a bound on the amount of such information, the authors managed to significantly improve the lower bound.

3.2.3 Symmetric Radio Networks

We consider here a restricted family of radio networks, where we assume that the underlying graph is *symmetric* (or equivalently *undirected*). Naturally, all algorithms designed for a directed radio network can be directly applied to an undirected radio network. As a result, our main focus here will be algorithms for broadcasting that can only be applied to undirected networks and obtain a faster execution time. For an undirected radio network, we exploit the fact that a node can be informed about the labels of the neighboring nodes (whereas in the directed model that is not necessarily true) and thus increase efficiency. The opposite holds for the lower bounds: a lower bound for a symmetric radio network consists a lower bound for arbitrary radio networks as well, whereas a lower bound obtained for general radio networks is not necessarily valid for the weaker model of undirected radio networks.

In undirected radio networks, we have the advantage that information about the network may travel in any direction. Thus, an important distinction is to consider *oblivious* and *adaptive* algorithms. As we have already mentioned, in oblivious algorithms, the decision of whether a node transmits at a given round depends only on the label of the node and the number of the round. On the other hand, in adaptive algorithms, the decision of whether a node transmits at a given round also depends on previously obtained messages. Another way to look at the difference between the two models is that in an oblivious algorithm all the actions of a node can be scheduled from the beginning, whereas an adaptive algorithm decides the actions of each node based on information obtained during the execution of the algorithm. An interesting study and comparison of both models was presented by Kowalski and Pelc in [56].

This section concentrates on adaptive deterministic algorithms. A linear-time algorithm (Explore-and-Expand) was first proposed in [13] . However, the authors assume that *spontaneous transmission* is allowed. This means that all nodes start the execution of the protocol simultaneously and a node can transmit without having received a message from the source node. This capability allows a preprocessing stage where the nodes exchange information before obtaining the source message and thus gain knowledge about the neighboring nodes.

The idea of the algorithm is pretty simple and is basically a distributed version of a *depth-first search algorithm* (DFS). However, the algorithm is not straightforward, as the size of the network n is assumed unknown to the nodes and thus it is not that simple for one node to learn its neighbors. The algorithm consists of several phases and each phase can be divided into 3 stages. Before we present the algorithm in detail, we need some definitions.

L_k : the set of nodes with labels $1, ..., 2^k$.

G_k : the connected component of the network containing the source and induced by L_k. If the label of the source is more than 2^k, G_k is the empty graph.

T_k : a spanning tree of G_k which is constructed in phase k.

C_k : an Eulerian cycle of T_k which is constructed in phase k.

$k - tag$: Nodes of C_k are labeled with consecutive positive integers, which are called k-tags.

Let us describe now in detail the stages of Explore-and-Expand at phase k.

STAGE A Stage A consists of 2^{k-1} rounds, which are numbered $2^{k-1}+1, \ldots, 2^{k-1}+2^{k-1}$. In round with number k, only the node with label k sends a message containing its label.

STAGE B Rounds are numbered $1, \ldots, 2^k$. Nodes in C_{k-1} that have received a message at stage A or earlier at stage B are called *active*. An active node sends a simple contact message in every round of stage B that corresponds to its k-tag.

STAGE C Stage C starts only when the source becomes active. Due to the previous stages, every node v knows its neighboring nodes N_v in G_k. Thus, it is possible to send a token to travel G_k in a DFS manner, while constructing C_k and assigning to the nodes of C_k with the appropriate k-tags. Initially, the source sends a message containing the token and a counter c set to 1. Node v executes the following procedure:

- Initialization : $Q_v = N_v$
- Message with `<visit>` : Node v removes from Q_v the neighbor that sent the message
- Message with counter c :

  $Q_v = \emptyset$:

- $v_{k-tag} = c + 1$.
- Node v sends the message `< label(v), visited>`
- Node v sends the message `<token, c+1, label(w)>` (w is the node from which v got the token)

$Q_v \neq \emptyset$:

- $v_{k-tag} = c + 1$.
- Node v sends the message `< label(v), visited>`
- Node v sends the message `<token, c+1, label(w)>` (w is the node in Q_v with the smallest label)

It can be proved that after phase k every node in G_k knows the source message. It remains to calculate how large k must be so that the message can reach all the nodes and that after phase k no messages are sent. Let $k : 2^{k-1} < n \leq 2^k$. Then, G_k is in fact the whole network and thus all nodes have received the message. Furthermore, since $G_r = G_{r+1}, r \geq k$, no node will transmit in stage A and B, and, as a result, no node will transmit in stage C. Considering the above, the algorithm stops after

$$\sum_{i=1}^{k}(2^{i-1} + 2^i + 2^{i+1}) = \sum_{i=1}^{k} 7 \cdot 2^{i-1} = 7(2^k - 1) = 14n - 1 = \mathcal{O}(n) \; rounds$$

Theorem 3.2.3. Explore-and-Expand *succeeds broadcasting in time $\mathcal{O}(n)$.*

As mentioned, the above algorithm assumes that a node can transmit spontaneously. Under this assumption, the algorithm is optimal, as proved in [56] by Kowalski and Pelc. To be more specific, the authors proved the following theorem for undirected networks:

Theorem 3.2.4. *For every integer n and every broadcasting algorithm there exists a network G with at most $2n+1$ nodes and diameter $D = 4$, such that the broadcasting time is $\Omega(n)$.*

Nevertheless, if we assume the harsher model with conditional wake-up, the fastest broadcasting algorithm was proposed by Kowalski and Pelc in [53] and completes broadcasting in $\mathcal{O}(n \log n)$ rounds. Before we describe the algorithm Select-and-Send, we first need to introduce two basic procedures. Let us assume that a node v knows a set of neighboring nodes A and a neighboring node $u \notin A$. Procedure Echo decides whether set A consists of 0, 1 or more nodes.

An important observation is that if $|A| = 1$, then node v will also know the label of the single node of set A. Let us assume now that a node v knows a neighbor u and that node v needs to select a node from a set S of its neighbors ($u \notin S$). Furthermore, we assume that v knows that some nodes in set S have labels at most m ($m = 2^k$). The procedure Binary-Selection can then be used to determine a single neighbor of set S in only $\mathcal{O}(\log m)$ rounds.

Procedure Echo(w,A)

```
(1) Every node in $A$ transmits its label ;
(2) Every node in $A \cap u$ transmits its label ;
switch messages received do
    case $v$ receives a message in step 1 and not in 2
        $|A| = 1$
    case $v$ receives a message in step 2 and not in step 1
        $|A| = 0$
    case $v$ receives no message
        $|A| > 1$
end
```

The main algorithm Select-and-Send can be described now in full detail. We define with *parent*(v) the node from which node v gets the token for the first time. Furthermore, when the token is at node v, S denotes the set of the neighboring nodes of v that have not been visited by the token.

Initial procedure : The source sends a message with the information that the neighboring node with label i should transmit in round $2i$. The source then waits for the first incoming message in round $2j$ (j is the smallest label of the neighboring nodes) and after that sends a message in round $2j+1$ in order to stop the procedure. Finally, the source sends the token to the node with label j.

Main procedure : (The token is at node v.) Initially, v sends the source message (all the neighbors receive it and wake up). Next, we distinguish two cases:

$v \neq source$ **:** Echo($parent(v)$, S)

$v = source$ **:** Echo(j, S)

According to the result of Echo the following happens:

$|S| = 0$ *and* $v \neq source$ **:** v sends the token to $parent(v)$ and stops.

$|S| = 0$ *and* $v = source$ **:** v stops.

$|S| = 1$ **:** v sends the token to the single node of S.

$|S| > 1$ **:** v If $v \neq source$ node v executes Echo($parent(v)$,$S \cap \{1, \ldots, 2^k\}$) (else Echo($j$,$S \cap \{1, \ldots, 2^k\}$) for $k = 1, \ldots$ and until $S \cap \{1, 2, \ldots, 2^k\}$ is not empty. Then, v executes Binary-Selection($parent(v)$, $S \cap \{1, \ldots 2^k\}$) and sends the token to the unique node that the procedure Binary-Selection chose.

It is clear that the initial procedure of algorithm Select-and-Send takes only $\mathcal{O}(n)$ rounds. Furthermore, we notice that during the main procedure of the algorithm, the

```
Procedure Binary-Select(u,m)
R := {1, . . . , m/2} ;
while true do
    (Let $R = \{x, \dots y\}$) ;
    Node $v$ transmits range $R$ ;
    Echo($v$, $R \cap S$) ;
    switch $|R \cap S|$ do
        case 1
            Node $v$ selects the single neighbor from $R \cap S$ ;
            break
        case 0
            $R := \{y + 1, \dots , y + (y - x + 1)/2\}$
        case > 1
            $R := \{x, \dots , (y + x - 1)/2\}$

    end
end
```

procedure Echo is called $\mathcal{O}(\log n)$ times from each node and thus Echo will be executed $\mathcal{O}(n \log n)$ times. As a result, the algorithm needs $\mathcal{O}(n \log n)$ rounds to terminate. The following theorem is straightforward.

Theorem 3.2.5. *Algorithm* Select-and-Send *performs broadcasting in an arbitrary undirected network in time* $\mathcal{O}(n \log n)$.

3.2.4 A quadratic broadcasting algorithm

Chlebus et al. [13] were the first to propose a quite simple broadcasting protocol that completes broadcasting in time $\mathcal{O}(n^2)$. The algorithm Simple-Sequencing works in a Round-Robin fashion. The algorithm is constructed so that at each round only one node transmits a message. Assuming that the size of the network n is known to the nodes, the implementation of the idea is pretty straightforward. The algorithm consists of n stages and each stage will have n rounds. In each stage, the node with label i transmits the message at round i of the stage. It is clear that the above procedure guarantees that the message will reach all nodes and that the running time is n^2. Nevertheless, if we assume no knowledge of the size n, things get more complicated. The final algorithm consists of phases, where *phase* k consists of 2^k segments and each segment of 2^k rounds. Algorithm 6 describes the execution of a *phase*. Note that phase k is basically an execution of 2^k Round-Robin procedures, restricted to nodes with label at most 2^k.

Algorithm Simple-Sequencing uses the *doubling technique*, i.e. the algorithm tests different values for the size of the network n by making exponential jumps. It is clear that

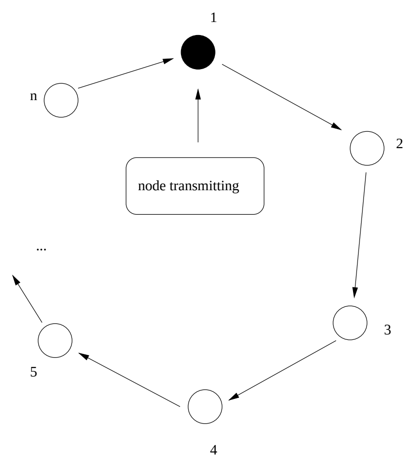


Figure 3.13: A graphical representation of Round-Robin

Algorithm 6: Simple-Sequencing

```
Node $v$ with label $i$ waits to receive the message ;
Assume that the message is received in round $r$ ;
$v$ transmits at the first round $r'$ $(r' > r)$ whose number in its segment is $i$.
```

each phase k lasts 4^k rounds. Let us now assume that $2^{l-1} \leq n < 2^l$. We can easily note that some node v that has distance d from the source node receives the message at the latest at segment d of phase l (that can happen when the label of node v is $> 2^{l-1}$ or the label of a node that the message must reach before node v is $> 2^{l-1}$). Thus, after phase l, all nodes have received the message and have transmitted once. Consequently, every node will stop transmitting after phase l. The total running time will thus be

$$\sum_{i=0}^{l} 4^k = \mathcal{O}(4^l) = \mathcal{O}(4^{\log n}) = \mathcal{O}(n^2). \tag{3.1}$$

By the above equation, we have the following theorem:

Theorem 3.2.6. *Algorithm* Simple-Sequencing *performs broadcasting in any network of size n in time $\mathcal{O}(n^2)$.*

3.2.5 The first subquadratic broadcasting algorithms

The question arising is whether we can improve the time complexity of Simple-Sequencing. The answer was given by Chlebus et al. in [13], where the authors proposed an algorithm with broadcasting in time $\mathcal{O}(n^{11/6})$, thus lowering for the first time the upper bound under quadratic time.

The algorithm is based on the notion of *selective families*, an essential tool in the field of radio networks. We have presented selective families in chapter 2; however, we have not mentioned the intuition behind the application of selective families to radio networks. The main difficulty in the setting of radio networks is the fact that collisions may occur when two nodes transmit simultaneously. Now, let us consider the set F of *active* nodes (i.e. nodes that have received the message). We would like to ensure that F will expand after some rounds of the algorithm, however this task is not simple as the nodes know nothing about the topology of the network. Nevertheless, if we could design a schedule that guarantees that after a number of rounds only one node of F transmits, that would mean that the message would reach some uninformed node without collision and thus F will grow larger. A *selective family* is a combinatorial tool that guarantees exactly the aforementioned condition.

The sets from a k-selective family $\mathcal{F}$ can be used as transmissions. We can map each round of a broadcasting algorithm to a specific set from $\mathcal{F}$ and then say that a node transmits in that round only when its label belongs to this specific set. Thus, two things should be designed by an algorithm that uses selective families. First, we need to find an appropriate mapping that ensures that broadcasting succeeds. We may need to consider a number of selective families for that case. Second, we need to construct such a selective family (or prove the existence of it). An important thing to keep in mind is that we are interested in small selective families, as these will lead to better execution times.

Under the setting of selective families, the algorithm Simple-Sequencing can be viewed as an algorithm that uses a simple selective family that consists of the singleton sets of $\{1, 2, \ldots, n\}$ and thus has size $\Omega(n)$. In order to lower the upper bound, we need to construct a smaller selective family with less than $\Omega(n)$ sets. For this, we will use the selective family $\mathcal{F}_m$ we introduced in section 2.2. Let $f_m = |\mathcal{F}_m|$ and $k_m = 2^{\lceil m/6 \rceil}$. Moreover, assume an enumeration for $\mathcal{F}_m$, $\mathcal{F}_m = \{F_1, F_2, \ldots, F_{f_m}\}$. We are now in position to describe the algorithm Selective-Broadcasting in full detail.

We define three different states for the nodes during the algorithm.

- **dormant / uninformed** : The node has not yet received the message
- **frontier / active** : The node has received the message, but has still uninformed neighbors.
- **passive / switched-off** : The node has received the message, all its neighbors are informed and it has stopped transmitting.

Let us also define that a node v is called m-node when there exists a directed path from the source to v such that all nodes of the path (the source and v also) have labels $\leq 2^m$. The algorithm is heavily based on the procedure Segment, which is described in detail.

Lemma 3.2.7. *After phase m of the algorithm* Selective-Broadcasting, *all m-nodes are switched off.*

Procedure SEGMENT(m)

```
HEAD :
for round $i = 1$ to $k_m \cdot f_m$ do
    An active node with label $k$ transmits when $k \in F_{i \bmod f_m}$
end
TAIL :
for round $i = 1$ to $2^m$ do
    An active node $v$ with label $k$ transmits at round $k$ ;
    Node $v$ becomes switched-off
end
```

Algorithm 8: SELECTIVE-BROADCASTING

```
for phase $i = 1$ to ... do
    call SEGMENT(i) $\lceil 2^i/k_i \rceil$ consecutive times
end
```

Proof. The proof is with induction to m. For $m = 0$, all 0-nodes are switched-off, since either the label of the source is larger than 1 and hence the set of $0 - nodes$ is empty, or else the source has label 1, so it transmits and then becomes switched-off.

Let us assume now that the lemma holds for $0 \leq m' \leq m - 1$. That implies that all $(m-1)$-nodes are switched off when phase m starts. We concentrate our attention in some call S of procedure SEGMENT. Denote by n_m the number of m-nodes still active or uninformed. We will prove that the number of the active nodes at the beginning of the tail of S is $\geq \min(k_m, n_m)$. If that holds, these active nodes will become switched-off during the tail of S. Since we have at most 2^m m-nodes, $\lceil 2^m/k_m \rceil$ calls of the procedure SEGMENT will be enough to get all nodes switched-off.

Let us prove now our hypothesis. We distinguish two cases. In the first case, the number of the active nodes at the beginning of the head of S is $\geq k_m$. This also holds at the end of the head of S and thus is straightforward. Otherwise, we will show that each *window* of f_m consecutive rounds increases by 1 the number of active m-nodes (until there exist no uninformed m-nodes). This implies directly that after k_m windows, we will have at least $\min(k_m, n_m)$ active nodes. Denote by A the set of active m-nodes that are adjacent to uninformed m-nodes at the beginning of a fixed window. Clearly, $|A| \leq k_m$. The case $A = \emptyset$ is trivial, as it implies that no uninformed nodes are left, thus $|A| > 0$. We can now apply the property of a k-selective family and note that at some round r of the window, a node $v \in A$ will be the sole transmitter. Fix an uninformed neighbor w of v. There are three subcases,

- w became active before round r: Then, we have one more active m-node.
- w is uninformed before round r and no other node transmits at round r : w becomes

active in round r

- w is uninformed before round r, but never gets the message because some other node w' transmits at the same time : Then, $w' \notin A$ and thus became active at some previous round of the window.

Thus, in each case the number of active nodes increases and the proof is concluded. □

Theorem 3.2.8. *In any network of size n, algorithm* Selective-Broadcasting *completes broadcasting in time* $\mathcal{O}(n^{11/6})$.

Proof. Let $m^* : 2^{m^*-1} < n \leq 2^{m^*}$. By lemma 3.2.7, after phase m^*, all nodes will become switched-off and no transmission will take place. Furthermore, we note that phase m needs $(k_m f_m + 2^m) \cdot \lceil 2^m/k_m \rceil$ rounds. The selective family we used has $k_m = \mathcal{O}(2^{m/6})$ and $f_m = \mathcal{O}(2^{5m/6})$. Substituting, we get that each phase needs $\mathcal{O}(2^{11m/6})$ rounds and by adding for phases $0, 1, \ldots, m^*$, we get that the algorithm needs $\mathcal{O}(2^{11m^*/6})$ rounds or $\mathcal{O}(n^{11/6})$ rounds (since $n = \mathcal{O}(2^{m^*})$). □

An important thing here is that algorithm Selective-Broadcasting can use any k-selective family. That implies that if we can find a smaller selective family, then the algorithm will run faster. Specifically, theorem 3.2.8 can be generalized as follows:

Theorem 3.2.9. *Let* $m^* : 2^{m^*-1} < n \leq 2^{m^*}$ *and consider a* k_m*-selective family* $\mathcal{F}_m$ *of* $[2^m]$ *with size* f_m*. In any network of size n, broadcasting can be performed in time*

$$\sum_{m=1}^{m^*} (k_m f_m + 2^m) \cdot \lceil 2^m/k_m \rceil. \tag{3.2}$$

De Marco and Pelc [62] used such a smaller selective family, namely the (Δ, n)-selective family of size $\mathcal{O}(\Delta^2 \log n)$ presented in section 2.3. Assuming that $\Delta = (2^m/m)^{1/3}$ and $n = 2^m$, we can construct a Δ-selective family of size $\mathcal{O}(2^{2m/3} \cdot m^{1/3})$. We can now apply theorem 3.2.9 with $k_m = \Delta = (2^m/m)^{1/3}$ and $f_m = \mathcal{O}(2^{2m/3} \cdot m^{1/3})$. After some simple calculations, we can deduce the following theorem:

Theorem 3.2.10. *In any network of size n, radio broadcasting can be completed in time* $\mathcal{O}(n^{5/3} \log^{1/3} n)$.

3.2.6 A broadcasting algorithm with running time $\mathcal{O}(n^{3/2})$

In [15], Chlebus et al. proposed several broadcasting algorithms that improved the best known upper bound of $\mathcal{O}(n^{5/3} \log^{1/3} n)$. The first algorithm performed broadcasting in time $\mathcal{O}(n^{1.77291})$, the second further improved the running time to $\mathcal{O}(n^{3/2}\sqrt{\log n})$. Finally, the third algorithm (Single-Prime) the authors proposed has running time only $\mathcal{O}(n^{3/2})$. In this section, we will only present the last algorithm with more details.

The idea behind the algorithm does not differ from previous ideas: we want to design a schedule in such a way that it guarantees that there will be rounds where only one node transmits. This time, the construction of the schedule is based on properties of prime numbers. Let us denote by p the smallest prime number that is greater or equal to $\lfloor\sqrt{n}\rfloor$, where n is the size of the network. Assuming that each node has a unique laebl from $[0,\dots,p^2-1]$, we can introduce the following mapping from $i \in [0,\dots,p^2-1]$ to a tuple:

$$i \longrightarrow < i\, div\, p,\ i\, mod\, p > \tag{3.3}$$

We call this tuple the *coordinates* of i. The set $\{0,\dots,p-1\}$ forms a field $\mathbb{F}_p$, since the arithmetic is modulo the prime p. We introduce the basic definition of a *line*.

Definition 3.2.11. *Let $a,b,c \in \mathbb{F}_p$ and $a > 0$ or $b > 0$. A **line** $L(a,b,c)$ is a set of nodes whose coordinates $< x,y >$ satisfy the following equation in $\mathbb{F}_p$:*

$$a \cdot x + b \cdot y = c \tag{3.4}$$

The definition of a line can lead us to some very interesting properties that will come handy afterwards.

Lemma 3.2.12. *The following properties hold:*

1. *Each line has size p.*
2. *The total number of lines is $p(p+1)$.*
3. *There are exactly $p-1$ lines disjoint with a given one. Two disjoint lines have the same direction.*
4. *Each node belongs to $p+1$ lines.*
5. *Two lines of different directions have exactly one common node.*
6. *For any two different nodes, there is exactly one line that contains both of them.*

The properties seem to have a lot in common with the properties of lines in the euclidean plane. The above properties are based to number theory, however they can also be viewed as geometric properties by mapping each node to a certain point on the euclidean plane. Then, a *line* as defined here would correspond to a geometric line on the plane. Figure 3.14 depicts the geometric intuition behind the aforementioned properties.

The idea behind the algorithm Single-Prime is to map each round to a specific line (or singleton), such that a node transmits in that round only if its coordinates belong to the line (or its label to the singleton). The use of lines and singletons is interleaved, so that we can use the advantages of both structures.

The following lemma is necessary in order to analyze the algorithm Single-Prime.

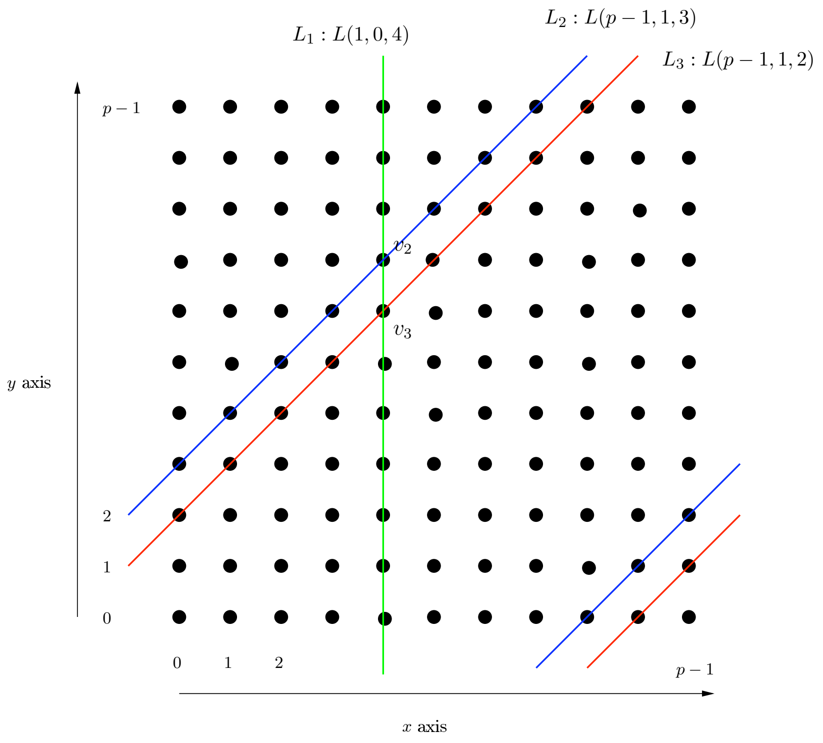


Figure 3.14: A geometric interpretation of the notion of the line.

Lemma 3.2.13. *Let F be a set of nodes and $|F| \leq p/2$. For each node $v \in F$ during each consecutive $2|F|$ stages, there are at least $|F|$ rounds, each round in a different stage, during which v is the only node from F that transmits.*

Proof. Since in each stage the lines used are disjoint, node v belongs to exactly one line of each stage. Let $T_1, T_2, \ldots, T_{2|F|}$ be these lines. Clearly, each line appears at most once in $2|F|$ consecutive stages and thus these lines have pairwise at most one common node (if they had two common nodes, it would be the same line). Consequently, the sets $T_1 - \{v\}, T_2 - \{v\}, \ldots, T_{2|F|} - \{v\}$ are disjoint. By the pigeonhole principle, this implies that at most $|F|-1$ of them include elements of F. Thus, at least $2|F|-(|F|-1) = |F|+1$ sets have no elements of F. □

Before we move to the proof of the basic lemma, let us first introduce an important term. Each time a node changes its status according to the following transitions:

- uninformed $\longrightarrow$ active.
- active $\longrightarrow$ passive.

Algorithm 9: SINGLE-PRIME

- In even-numbered rounds, singletons are used. During each consecutive p^2 even rounds, each singleton is used exactly once, and thus each node transmits exactly once.
- In odd-numbered rounds, lines are used. During each consecutive $p(p+1)$ odd rounds, each line is used exactly once. Furthermore, lines of the same direction are used in consecutive odd rounds. We define a *stage* as $2(p+1)$ consecutive rounds where we use only lines of the same direction.

we say that *progress* has been made. Each transition contributes a unit to the measure of progress. When progress reaches $(2n-1)$ broadcasting is complete (notice that the source node is from the beginning active).

We can now prove the basic lemma, which gives as a lower bound to the amortized progress of the algorithm SINGLE-PRIME.

Lemma 3.2.14. *Let F be the set of active nodes at the beginning of stage s and assume that $|F| \leq p/2$. Then, the average progress per stage from stage s to stage $s+2|F|-1$ is $\Omega(1)$.*

Proof. Let us consider the $2|F|$ consecutive stages $T = \{s, s+1, \ldots, s+2|F|-1\}$. If each of the nodes in F transmits during T as the only active node, then we make progress at least $|F|$ and thus the average progress per stage is at least $1/2$. Assume now that some node v in F never transmits as the only active node. However, by lemma 3.2.13, there are at least $|F|$ rounds where v transmits as the only node in F. Thus, some node $v' \neq v$ transmits simultaneously with v. Each line is uses at most once in T, so any two nodes transmits at most once during T. Consequently, if v transmits along with other nodes during $|F|$ stages, then these nodes are distinct. The latter implies that at least F nodes became active since the beginning of T and so we achieve again an average progress of at least $1/2$. □

The above lemma guarantees progress when the set of active nodes is $\leq p/2$. When the active nodes are more than $p/2$, the round-robin ensures that we make enough progress. Thus, the two interleaved algorithms complement each other and putting them together, we can achieve the $\mathcal{O}(n^{1/2})$ running time.

Theorem 3.2.15. *The algorithm* SINGLE-PRIME *completes broadcasting in time $\mathcal{O}(n^{1/2})$.*

Proof. We have to calculate the time needed to make progress $2n-1$. If at some stage the number of active nodes k is less than $p/2$, by lemma 3.2.14, we have a constant average progress per stage over the next $2k$ stages and thus the average progress per round is

$\Omega(1/p) = \Omega(n^{-1/2})$. If now the number of active nodes is greater than $p/2$, the round-robin algorithm guarantees progress $p/2 = \Omega(n^{-1/2})$ over the next $\mathcal{O}(n)$ rounds, and thus again average progress per round $\Omega(\sqrt(n)/n) = \Omega(n^{-1/2})$. This implies that we need $\mathcal{O}(n/n^{-1/2}) = \mathcal{O}(n^{3/2})$ rounds to complete broadcasting. □

3.2.7 Broadcasting in time dependent on Δ

Clementi et al. [23] proposed a new broadcast technique that yields broadcasting protocols with execution time that does not contain the parameter n as linear factor, but instead contains the parameters D and Δ. Specifically, the authors obtained a $\mathcal{O}(D\Delta \log(n/\Delta) \log^{1+a} n)$ running time, for any positive constant a. The key concept of the algorithm is the use of selective families. More precisely, we use (n, Δ)-selective families.

In order to provide the intuition behind the use of selective families, let us first describe a naive algorithm, which we will then modify to obtain the final algorithm. Consider a radio network with size n and maximum in-degree Δ and let $\mathcal{F} = \{F_1, \ldots, F_m\}$ be a (n, Δ)-selective family. We now apply the selective family in the usual way, i.e. a node with label x transmits at round i of and only if $x \in F_i$. This single application of the selective family (which requires m rounds) guarantees that for each set of $d \leq \Delta$ nodes, at some round only one of the nodes of the set transmits. As a result, at least one uninformed node will be informed. Clearly, by repeating the above procedure n times, all the nodes of the network will receive the message. Thus, we have achieved broadcasting in time $n|\mathcal{F}|$.

As we have seen, there exist (n, Δ)-selective families with size $\mathcal{O}(\Delta \log(n/\Delta))$. Thus, we can assume that $|\mathcal{F}| = \mathcal{O}(\Delta \log(n/\Delta))$. The next step is to show how the above naive algorithm can be modified to perform more efficiently. We will first present an algorithm Broad-A(n, Δ) that works with the assumption that n and Δ are known to the nodes and then we will remove this assumption.

Algorithm 10: Broad-A(n, Δ)

At the first round, the source transmits and then becomes non-active. Then, the algorithm performs consecutive identical *stages*. At round j of stage i, node v transmits only if the following two conditions hold:

- the label of v belongs to F_j.
- v received the message for the first time at stage $i-1$, i.e. the previous stage.

After transmitting, node v becomes switched-off and stops transmitting.

The following theorem holds for the running time of algorithm Broad-A.

Theorem 3.2.16. *For any network with size n, eccentricity D and maximum in-degree Δ, algorithm* Broad-A(n, Δ) *performs broadcasting in time $\mathcal{O}(D\Delta \log(n/\Delta))$*

Proof. As we have mentioned above, $|\mathcal{F}| = \mathcal{O}(\Delta \log(n/\Delta))$ and thus each stage lasts $\mathcal{O}(\Delta \log(n/\Delta))$ rounds. Thus, it is enough to prove that we need $D-1$ stages. More precisely, we will show that a node v receives the message at stage i if and only if v is at distance $i+1$ from the source. We will use induction. For $i = 0$, the claim holds trivially. Assume now that all nodes at distance i have received the message for the first time at stage $i-1$. Consider some node w at distance $i+1$ from the source. Node w has at least one in-neighbor at distance i, which will be informed. Furthermore, following the second condition necessary for a node to transmit, only nodes at distance i transmit at stage i. Denote by S the set of in-neighbors of w at distance $i-1$. Clearly, $|S| \leq \Delta$ and thus, by the selectivity property of the family $\mathcal{F}$, at some round of stage i only one node of S transmits and w gets the message. □

Next, we drop the assumption that Δ is known to the nodes and assume only knowledge of n. We use the *doubling technique* in order to perform broadcasting. The new algorithm Broad-A(n) runs a sequence of stages, where each stage has $\lceil \log n \rceil$ rounds. In round r of stage s, each node runs round s of Broad-A$(n, 2^r)$. Thus, the algorithm tries in a way to search for the apprpriate value of the parameter Δ. The next theorem is naturally obtained.

Theorem 3.2.17. *For any network with size n, eccentricity D and maximum in-degree Δ, algorithm* Broad-A(n) *performs broadcasting in time $\mathcal{O}(D\Delta \log(n/\Delta) \log n)$*

The last step is to remove the assumption that the nodes know n. The final algorithm basically consists of executions of Broad-A(2^l) for $l = 1, 2, \ldots$. However, since a simple *dovetailing* technique results in unsatisfying performance, the authors proposed a more sophisticated dovetailing procedure that adds only a factor of $\mathcal{O}(\log^a n)$, for $a > 1$. Thus, we obtain the following result for the final algorithm Broad-A^a.

Theorem 3.2.18. *For any positive constant a, algorithm* Broad-A^a *performs broadcasting in time $\mathcal{O}(D\Delta \log(n/\Delta) \log^{1+a} n)$ in any network with size n, eccentricity D and maximum in-degree Δ.*

3.2.8 An almost optimal broadcasting algorithm

In [18], Chrobak et al. proved an upper bound of $\mathcal{O}(n \log^2 n)$ for radio broadcasting, which is only a logarithmic factor away from the lower bound. The authors, however, proved only the existence of an algorithm with this running time and thus there is no explicit construction of the algorithm.

The authors used k-selectors for the broadcasting algorithm. The intuition behind the use of selectors is that we want to have a guarantee that there will be a round where only one active node transmits, but we also want to ensure that the transmission will not be blocked by nodes that have recently become active. As we have mentioned in chapter 2, for each n and positive integer $w \leq n$, we can prove the existence of w-selectors S with size $\mathcal{O}(w \log n)$.

For each $i \in \{0, \dots \log n\}$, define by $S_i = \{S_{i,0}, S_{i,1}, \dots, S_{m_i-1}\}$ a 2^i-selector with size $m_i = \mathcal{O}(2^i \log n)$. The algorithm SELECTOR-BROADCAST is described below.

Algorithm 11: SELECTOR-BROADCAST

```
for stage $s = 1$ to ... do
    for round $i = 1$ to $\log n + 1$ do
        All active nodes with label $l$ and $l \in S_{i,s \bmod m_i}$ transmit
    end
end
```

Theorem 3.2.19. *The algorithm* SELECTOR-BROADCAST *completes broadcasting in time* $\mathcal{O}(n \log^2 n)$.

Proof. The proof of the running time of the algorithm is pretty similar to the proof in [15] for the algorithm SINGLE-PRIME. We will use amortized analysis and prove that for every stage s, there is some stage $s' > s$, such that the average progress per stage is $\Omega(1/\log n)$. Each stage has $\mathcal{O}(\log n)$ rounds and the total progress needed to complete broadcast is $2n - 1$. Thus, we conclude then that the total rounds needed are $\mathcal{O}(n/(1/\log^2 n)) = \mathcal{O}(n \log^2 n)$.

Assume that F is the set of active nodes at the beginning of stage s. Let $i : 2^{i-1} \leq |F| \leq 2^i$. Furthermore, let us denote by Y_i the set of nodes that first received the message in some of the stages $s, s+1, \dots, s+m_i-1$. We can distinguish two cases:

Case A : There is some i such that $|Y_i| \geq 2^i$. Then, after m_i stages the progress is $|Y_i| \geq 2^i$ and thus the average progress for each stage is $\Omega(2^i/m_i) = \Omega(1/\log n)$.

Case B : For every i, we have that $|Y_i| \leq 2^i$. Let us now consider some node v that has not yet received the message at the beginning of stage s and that has some active neighbor. Denote by X the set of its active neighbors at the beginning of stage s and assume some $i : 2^{i-1} \leq |X| \leq 2^i$. Clearly, it also holds that $|Y_i| \leq 2^i$. Then, by the definition of a selector, there exists some set $S_{i,r} \in S_i$ such that the set $S_{i,r}$ hits X and avoids Y_i. By the construction of the algorithm, that set will be the transmission set in one of the stages $s, \dots, s+m_i-1$.

In the round where the set $S_{i,r}$ is used the following happen:

- $|S_{i,r} \cap X| = 1$ and thus exactly one neighbor of v transmits.
- $|S_{i,r} \cap Y_i| = 0$ and thus no node from Y_i transmits.
- The rest of the passive nodes remain passive during the stages $s, \dots, s+m_i-1$ and so do not transmit.

Figure 3.15 shows schematically the analysis stated. The above implies that only one active neighbor of v transmits and thus v receives the message. Consequently, all the nodes in F receive the message before stage $s + m_f - 1$ and the average progress will again be $\Omega(1/\log n)$. $\square$

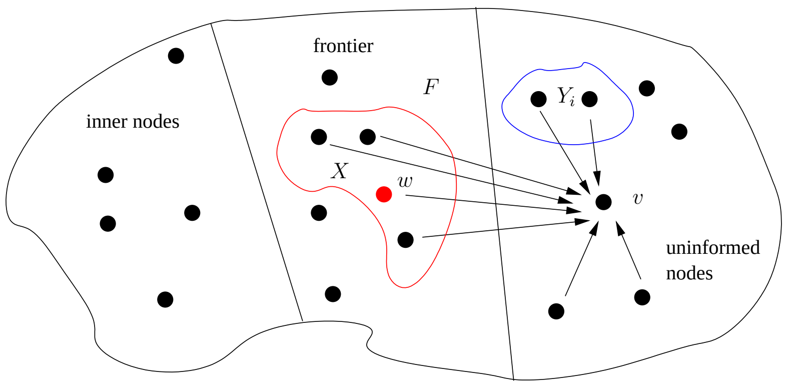


Figure 3.15: The analysis for the case B of the proof. Node w (red color) is the only node transmitting from the set F of active nodes. The nodes of Y_i remain silent. The rest of the uninformed nodes are not yet awakened and thus do not transmit. In the end, w transmits, no collision happens and thus v receives the message.

3.2.9 Recent improvements on broadcasting time

In [54], Kowalski and Pelc constructed an algorithm that completed broadcasting in time $\mathcal{O}(n \log n \log D)$ for n-size networks of eccentricity D. The algorithm improves the bound for broadcasting for radio networks that have eccentricity polylogarithmic in n. Nevertheless, the authors assumed that the nodes know a linear bound on the size of the network. If we assume that the size is unknown, the algorithm can be modified and succeed broadcasting with running time $\mathcal{O}(n \log n \log\log n \log D)$. The algorithm is again non-constructive, as the authors proved only the existence of a combinatorial tool that the nodes used during the broadcasting. The combinatorial tool used is a random boolean matrix, which is used as a transmission set. The analysis is quite complicated and will not be presented here (for more details see [54]).

Instead, we will present an algorithm presented by Czumaj and Rytter [25], which further improves the running time of broadcasting to $\mathcal{O}(n \log^2 D)$. Their approach is based on a combination of infinite *selecting sequences* and *selectors*. First, we will need to introduce some necessary definitions.

Definition 3.2.20. *Consider an infinite sequence $\mathcal{J}$ from the set $\{0, 1, \ldots, \log D\}$.*

- *$sparseness(\mathcal{J}, k)$ is the minimal distance between two positions in $\mathcal{J}$ containing k.*

- *density*$(\mathcal{J}, k)$ *is the smallest integer* l*, such that in every subsequence of* $\mathcal{J}$ *of* l *consecutive elements,* k *appears at least once.*

Upon the above measures, we can define two properties of an infinite sequence $\mathcal{J}$.

Definition 3.2.21. *Consider an infinite sequence* $\mathcal{J}$ *from the set* $\{0, 1, \dots, \log D\}$.

D-sparseness: *sparseness*$(\mathcal{J}, k) = \Omega(2^k)$*, for each* $0 \le k \le \log D$.

D-density: *density*$(\mathcal{J}, k) = \mathcal{O}(2^k)$*, for each* $0 \le k \le \log D$.

We want to construct a sequence $\mathcal{J}$ that satisfies both the properties of *D-sparseness* and *D-density.* Luckily, such a sequence can be constructed in a simple way. We define first a finite sequence J of length $2D - 1$ with the property that for any $i, 1 \le i \le 2D - 1$

$$J_i = k \Leftrightarrow (i \mod 2^{k+1}) = 2^k$$

The infinite sequence $\mathcal{J}$ occurs by repeating the sequence J. It is easy to see that $\mathcal{J}$ satisfies both properties. For better understanding, let us provide an example of such a sequence for $D = 8$.

$$\mathcal{J} =< 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0 | 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0, \dots >$$

As far as selectors is concerned, we will need a j-selector $\mathcal{F}$ of size $\mathcal{O}(j \log(n/j))$, the existence of which has been proved via the probabilistic method. Denote by $\mathcal{F}^j = \{\mathcal{F}_1^j, \mathcal{F}_2^j, \dots, \mathcal{F}_{s_j}^j\}$ a 2^j-selector of size $s_j = \mathcal{O}(2^j \log(n/2^j))$, where $0 \le j \le \log n$. We can now describe the algorithm ObliviousBroadcast in full detail.

Algorithm 12: ObliviousBroadcast(M)

```
for $t = 1$ to $M$ do
    $k = \mathcal{J}_t$ ;
    (Phase number $t$ of rank $k$)
    for $j = 1$ to $\log(2^k n/D)$ do
        apply the selector $\mathcal{F}^j$
    end
end
```

We should notice that algorithm ObliviousBroadcast is *oblivious*, i.e. each node knows from the beginning the sequence of transmissions. Let us now proceed to the analysis. The analysis concentrates on a single path P from the source to an arbitrary node and we will calculate the value of M, such that all nodes of the path receive the message. We consider the *layers* of the path $P =< v_0, v_1, \dots, v_l >$, where each layer $L_P(i)$ is the set of all the in-neighbors of v_i which are not in-neighbors for any node $v_j : j > i$. We call a layer *leading* when it is the highest ranking layer with an active node. We first prove the following lemma.

Lemma 3.2.22. *If $|L_P(i)| \leq 2^k \cdot (n/D)$ and $L_P(i)$ is leading, after a phase of rank k, $L_P(i)$ is no longer leading.*

Proof. The proof is based on the properties of a selector. Denote by $sub - phase(k, j)$ the part of a phase of rank k where the selector $\mathcal{F}^j$ is applied. Also, denote by X the set of nodes of $L_P(i)$ that are active at the beginning of *subphase*(k, j) and by Y the set of nodes of $L_P(i)$ that will be activated during the sub-phase. Let an integer j, where $2^{j-1} < |X| < 2^j$. We distinguish two cases:

- If $|Y| \leq 2^j$, the properties of a 2^j-selector guarantee that only a single node of X transmits, thus $L_P(i)$ will no more be the leading layer after the sub-phase.
- If $|Y| > 2^j$, that means that the number of active nodes doubles after the sub-phase. Since that cannot happen in every sub-phase (due to the constraint on the size of the layer), at some sub-phase the layer $L_P(i)$ will stop leading.

□

Let us now try to estimate the phases when layer $L_P(i)$ is leading. Due to the above lemma and the *D-density* property of $\mathcal{J}$, layer $L_P(i)$ is leading for $\mathcal{O}(|L_P(i)|(D/n) + 1)$ phases. Summing up, the number of phases in order to reach the last layer will be $\mathcal{O}(\sum_{i=1}^{l} |L_P(i)|(D/n) + 1) = \mathcal{O}(D)$. Thus, we have proved the following lemma.

Lemma 3.2.23. *There exists a constant c such that if $M \geq cD$,* ObliviousBroadcast*(M) completes broadcasting.*

We are now in position to prove the main theorem. At first, we can easily show that a phase with rank k requires $\mathcal{O}(2^k \cdot (n/D) \cdot \log D)$ rounds. Using the above lemma and the *D-sparseness* of sequence $\mathcal{J}$, notice that we run $\mathcal{O}(D/2^k)$ phases of rank k. Now, by adding the time needed for all ranks, we have that the total number of rounds is

$$\mathcal{O}\left(\sum_{k=0}^{\log D} (2^k \cdot (n/D)) \cdot \log D) \cdot (D/2^k)\right) = \mathcal{O}\left(\sum_{k=0}^{\log D} n \log D\right) = \mathcal{O}(n \log^2 D)$$

Thus, the following theorem is proved.

Theorem 3.2.24. ObliviousBroadcast *completes broadcasting in an arbitrary network of size n and eccentricity D in $\mathcal{O}(n \log^2 D)$ rounds.*

ObliviousBroadcast performs the same as Selector-Broadcast from [18] in the worst case. Very recently, De Marco [61] presented a deterministic algorithm that performs broadcasting in $\mathcal{O}(n \log n \log \log n)$ time, which clearly consists an improvement in any case. The algorithm is only a factor of $\log \log n$ from the upper bound for distributed deterministic broadcasting. It also assumes that the nodes are only aware if their own label. In this section we will only provide the outline and the ideas behind the algorithm.

The algorithm is non-constructive, as all the nodes of the network use during the execution a combinatorial object whose existence is proved through the probabilistic method. The combinatorial object used is called a *transmission schedule*. The transmission schedule is a matrix $\mathcal{T}$, whose entries $\mathcal{T}_{i,j}$ are subsets of $\{1, 2, \ldots, n\}$. Each entry of the matrix then corresponds to a transmission set used in the algorithm. The columns of $\mathcal{T}$ correspond to time slots, which means that some node v transmits at round j if and only if $v \in \mathcal{T}_{i,j}$ for some i.

Now, each node starts from the first row of a column k, where k is the round when it first receives the message. Then, it moves to the second row and the procedure continues in that way. A property of the matrix is that the number of the elements of the transmission sets decreases as the row number increases. Thus, a node starts transmitting with high frequency when receiving a message and by the time decreases the transmission rate. The demanding part is to construct such a matrix with properties that ensure the upper bound on the time complexity.

3.3 Randomized Broadcasting

Randomization has proved to be a very effective tool in the setting of radio networks, particularly when it comes to distributed broadcasting protocols. The use of randomization helps us overcome the fact that the nodes know nothing about the topology of the network and that coordination between the nodes proves to be very difficult and expensive. In fact, randomized protocols can in some cases perform broadcasting exponentially faster than deterministic protocols, which shows how powerful randomization can be.

Let us first state a few things about the setting of the broadcasting algorithms that will be presented in this section. We assume the classical ad-hoc radio network, where the nodes know only their own label (sometimes even that assumption can be removed). It is also important to note that we do not have to distinguish between directed and undirected radio networks, as most randomized algorithms can be deployed to both cases without many modifications.

First, let us examine the lower bounds that have been stated for randomized algorithms. As we have seen in a previous section, Alon et al. [1] has shown that there exists a family of networks of size n and eccentricity 2 for which any broadcasting algorithm requires at least $\Omega(\log^2 n)$ rounds to complete broadcasting. This bound holds not only for deterministic, but also for randomized protocols. Kushilevitz and Mansour [58] also proved that any randomized algorithm has expected running time $\Omega(D \log(n/D))$, where D is the eccentricity of the network. By combining the two lower bounds, we come up with a $\Omega(D \log(n/D) + \log^2 n)$ lower bound for randomized broadcasting.

The first randomized algorithm for broadcasting was proposed by Bar-Yehuda et al. in [2] and is almost optimal, with expected execution time $\mathcal{O}(D \log n + \log^2 n)$. This algorithm is optimal when $D \leq n^{1-\epsilon}$, but when D is close to n, the algorithm requires $\mathcal{O}(n \log n)$ time and thus is a logarithmic factor above the lower bound. In [53], Kowalski and Pelc improved the upper bound by presenting an algorithm that performs broadcasting in expected time $\mathcal{O}(D \log(n/D) + \log^2 n)$. Around the same time, Czumaj and Rytter in [25] provided independently another randomized algorithm that also performed broadcasting in expected time $\mathcal{O}(D \log(n/D) + \log^2 n)$ with high probability.

3.3.1 A lower bound for randomized broadcasting

Kushilevitz and Mansour [58] were the first to prove a $\Omega(D \log(n/D))$ lower bound for the expected running time of a randomized algorithm. The authors proved that, for any Las Vegas algorithm (i.e. the error probability is zero) and any parameters n and D, we can construct an ordering of the n nodes in a network of eccentricity D, such that the expected number of rounds used by the algorithm is $\Omega(D \log(n/D))$. The authors distinguished two different cases of distributed randomized algorithms:

uniform algorithm : All nodes run exactly the same program.

non-uniform algorithm : Each node has a different program.

It is clear that the *uniform* case is simpler to analyze. Therefore, the authors first deal with the uniform case. They consider a family of networks $\mathcal{C}_n$ with $n+1$ nodes, which is composed of a clique of size n and a source node connected to t, $t \leq n$ of the nodes in the clique. For any broadcasting protocol Π, let us denote by $E\left[T_t^{\Pi}\right]$ the expected number of rounds until the first *successful* round (i.e. a round where exactly one of the t nodes transmits). The following lemma can be proved.

Lemma 3.3.1. *Denote by E_t the expectation when t is chosen uniformly from the set $\{2^0, 2^1, \ldots 2^{\lfloor \log n \rfloor}\}$. Then,*

$$E_t\left[E\left[T_t^{\Pi}\right]\right] = \Omega(\log n)$$

For the proof of the main result, they consider a family of networks $\mathcal{G}_{n,D}$. Let $l = n/D$ and assume for simplicity that l is a power of 2. A network in $\mathcal{G}_{n,D}$ is a complete layered network consisting of $D+2$ layers. The first layer contains the source node, while each of the next D layers consists of $l_i = 2^{t_i}$ nodes, where t_i is chosen uniformly and independently for each layer in the range $1, \ldots, \log l$. The last layer contains all the other nodes. As for the connections between the nodes, each node in layer i is connected to all nodes in layers $i-1$ and $i+1$.

The important thing about that construction is that the broadcasting progresses in a layer-by-layer fashion. When a node in layer i receives the message, all the other nodes of layer i will receive the message at the same round. This means that nodes of the same layer will share the same history and transmit with the same probability distribution. Thus, lemma 3.3.1 can be directly deployed. Using this idea, the theorem can now be derived.

Theorem 3.3.2. *For any uniform broadcast protocol, for every number of nodes n and eccentricity D, there exists a network in which the expected time to complete broadcasting is $\Omega(D \log(n/D))$.*

As for the non-uniform case, it can be reduced to the uniform case and, as a result, theorem 3.3.2 also holds for the non-uniform case. The above theorem can also be extended to include protocols that err with a small probability (Monte Carlo algorithms).

Apart from Kushilevitz and Mansour, Liu and Prabhakaran [59] also proposed a proof for the lower bound, using a different construction. More specifically, they proved the following theorem:

Theorem 3.3.3. *For any Monte Carlo broadcasting algorithm, there is a network with n nodes and eccentricity D, in which the algorithm has $\Omega(D \log(n/D))$ expected running time.*

The authors used the *Yao's minimax principle* to reduce the task of proving a randomized lower bound to that of proving a deterministic lower bound. We describe Yao's minimax principle, as described in [63].

Theorem 3.3.4 (**Yao's Minimax Principle**)**.** *Let Π a problem with a finite set $\mathcal{I}$ of input instances and a finite set of deterministic algorithms $\mathcal{A}$. Let $C(I, A)$ be the running time of algorithm A in input I. For a probability distribution $\mathbf{p}$ over $\mathcal{I}$ and $\mathbf{q}$ over $\mathcal{A}$, let $I_{\mathbf{p}}$ denote a random input chosen according to $\mathbf{p}$ and $A_{\mathbf{q}}$ a random algorithm chosen according to $\mathbf{q}$. Then, for all distributions $\mathbf{p}$ and $\mathbf{q}$*

$$\min_{A \in \mathcal{A}} \mathbf{E}[C(I_{\mathbf{p}}, A)] \leq \max_{I \in \mathcal{I}} \mathbf{E}[C(I, A_{\mathbf{q}})]$$

This theorem suggests that the expected running time for the fastest deterministic algorithm on a random input is a *lower bound* for the expected running time of the optimal Las Vegas algorithm for the problem. Thus, instead of providing directly a lower bound for randomized algorithms, we can choose an appropriate probability distribution $\mathbf{p}$ and prove a bound for the expected running time of deterministic algorithms on $I_{\mathbf{p}}$.

However, the above theorem is stated for Las Vegas algorithms, whereas we are interested in Monte Carlo algorithms with error probability $\epsilon \in [0, 1/2]$. We denote now with $C_\epsilon(I, A)$ the running time of algorithm A that errs with probability ϵ on input I. Then, we get an analogous proposition to Yao's minimax theorem.

Proposition 3.3.5. *For all distributions $\mathbf{p}$ and $\mathbf{q}$ and any $\epsilon \in [0, 1/2]$:*

$$\frac{1}{2} \min_{A \in \mathcal{A}} \mathbf{E}[C_{2\epsilon}(I_{\mathbf{p}}, A)] \leq \max_{I \in \mathcal{I}} \mathbf{E}[C_\epsilon(I, A_{\mathbf{q}})]$$

We prove the lower bound for a family $\mathcal{F}_{n,D}$ of networks with $\Theta(n)$ nodes and $\Theta(D)$ eccentricity, which we construct explicitly. First, let us define another family of networks $\mathcal{D}_m$. A network of the family $\mathcal{D}_m$ consists of a source, a destination and m intermediate

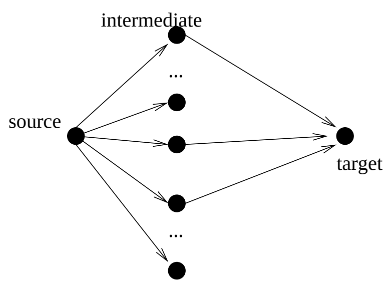


Figure 3.16: A graph of the family $\mathcal{D}_m$

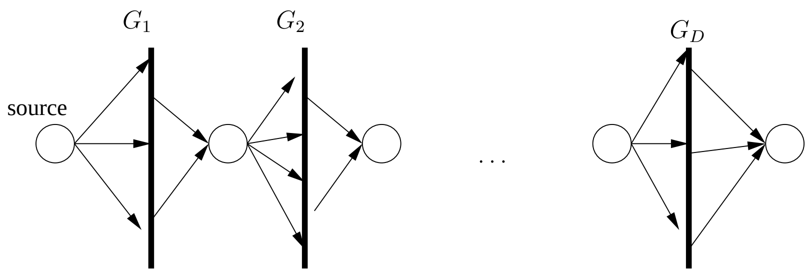


Figure 3.17: A graph of the family $\mathcal{F}_{n,D}$

nodes. The source is connected with all the intermediate nodes, whereas only some of the intermediate nodes are connected with the destination (see figure 3.16). A network of the family $\mathcal{F}_{n,D}$ consists of D layers G_i, where each layer is a network $\mathcal{D}_{\lfloor n/D \rfloor}$ (see figure 3.17).

It can be proved that there exists a probability distribution $\mathbf{P}$ over $\mathcal{D}_m$, such that the probability for the destination to get the message in any one deterministic step is $\mathcal{O}(1/\log m)$. Now, let us consider a deterministic broadcasting algorithm for $\mathcal{D}_m$ that errs with probability $1-p$, where $p = \Omega(1)$. Let t_{min} be the smallest number so that for a fraction $p/2$ of the inputs, the algorithm completes broadcasting in $\leq t_{min}$ steps. Then, for $\geq p/2$ of the fraction, the algorithm needs at least t_{min} steps ($p \leq 1$). That means that the expected running time is at least $p/2 \cdot t_{min}$. However, t_{min} must be $t_{min} = \Omega(1/(1/\log m)p/2) = \Omega(\log m)$ and thus the expected running time will also be $\Omega(\log m)$.

Next, we need to extend this result to the family $\mathcal{F}_{n,D}$. The probability distribution for $\mathcal{F}_{n,D}$, $\mathbf{P}^*$ occurs by independently choosing the probability distribution for each layer G_i. Intuitively, as we have D layers and by linearity of expectation, we get an expected running time of $D \cdot \Omega(\log m) = \Omega(D\log(n/D))$. We will not present here the technical issues that need to be addressed in order to formally prove the above.

We have now proven that any deterministic algorithm with error probability $1-\Omega(1)$ has over $\mathbf{P}^*$ expected running time $\Omega(D\log(n/D))$. By applying Yao's minimax theorem, we conclude that the lower bound for randomized algorithms that err with probability

$\frac{1}{2} - \Omega(1)$. However, we can make the error probability arbitrarily small by running the algorithm a constant number of times and thus the lower bound has been established. Last, let us note that the lower bound also holds for undirected radio networks, under the assumption that no spontaneous transmissions are permitted.

3.3.2 Randomized algorithms for broadcasting

The first randomized algorithm for broadcasting in radio networks was presented by Bar-Yehuda et al. [2] and manages to complete broadcasting in almost optimal running time. The algorithm is based on a very simple procedure (Decay), which uses randomization in order to resolve the conflicts that may appear when many neighboring nodes transmit at the same round. The basic difficulty in radio networks is that coordination between neighboring nodes is necessary to avoid conflicts and that coordination is highly inefficient in a distributed setting. Randomization can help us overcome this difficulty.

The idea behind the procedure Decay is the following. Suppose that a group of nodes wishes to transmit a message to the same target node. It is clear that the message will be received only if exactly one node transmits the message at a given round. Thus, after transmitting a message, each node stops transmitting with probability 0.5 (flips a fair coin). As a result, after each round, approximately half nodes continue transmitting.

```
Procedure DECAY($k$,$m$)
  $i := 0$ ;
  repeat
    transmit message m ;
    flip coin ;
    $i++$ ;
  until $coin = head$ or $i > k$ ;
```

Before we describe the algorithm in full detail, we provide an important lemma concerning the procedure Decay.

Lemma 3.3.6. *Let v be a node of the network G. Also, let d ($d \geq 2$) neighbors of v that start the execution of procedure* Decay(k, m) *at round 0. Denote with $Pr[k, d]$ the probability that v receives message m by round k. Then*

1. $\lim_{k\to\infty} Pr[k, d] \geq 2/3$
2. $k \geq 2\lceil \log d \rceil \Rightarrow Pr[k, d] > 1/2$

The lemma can be proved using elementary probabilistic arguments and thus the proof will be omitted. Procedure Decay is the basis of the main algorithm Broadcast, however we can not use it straightforward. The number of the neighbors in not known to the nodes and thus we must use an upper bound on the possible neighbors Δ (where Δ is the maximum indegree of a node). We must also ensure that each node starts procedure Decay at the exact same round and executes the procedure enough times for the message to have a high probability to reach every node of the network. Finally, we assume that the nodes are not aware of the actual size of the network, but only of a linear upper bound N. Each process executes the procedure Broadcast, but for the algorithm to begin we also need that the source node to transmit message m.

Procedure Broadcast(Δ, N)

```
$k := 2\lceil \log \Delta \rceil$ ;
wait until receiving message $m$ ;
for $2\lceil \log(N/\epsilon) \rceil$ times do
    while round mod $k <> 0$ do wait ;
    Decay $(k, m)$ ;
end
```

Let us now provide some analysis of the above algorithm. First, let $M(\epsilon) = \sqrt{\log(n/\epsilon)}$ and $T(\epsilon) = 2D+5M(\epsilon)\max(\sqrt{D}, M(\epsilon))$. Let us also assume that the main loop of Broadcast runs infinitely and let T_v be the random variable denoting the round when node v receives message m. Also, let $T_f = \max_v T_v$. It is clear that T_f is a random variable denoting the round when the last node receives a message. The following lemma holds.

Lemma 3.3.7. *For all* $0 < \epsilon \leq 1$

1. $Pr[T_v > 2\lceil \log \Delta \rceil \cdot T(\epsilon)] < \epsilon/n$
2. $Pr[T_f \leq 2\lceil \log \Delta \rceil \cdot T(\epsilon)] > 1 - \epsilon$

Proof. Let D_i be the length of a shortest path from the nodes that have already received m at phase i to the node v (a phase is $2\lceil \log \Delta \rceil$ consecutive rounds). For the first phase, it is clear that $D_0 \leq D$. Next, we will try to bound the probability $Pr[D_{T(\epsilon)} > 0]$, which is in fact the probability that that node v has not received the message by phase $T(\epsilon)$ (i.e. round $2\lceil \log \Delta \rceil \cdot T(\epsilon)$)

$$Pr[D_{T(\epsilon)} > 0] = Pr[\sum_{i=0}^{T(\epsilon)-1} (D_i - D_{i+1}) < D_0] \leq Pr[\sum_{i=0}^{T(\epsilon)-1} (D_i - D_{i+1}) < D]$$

Let $X_i = D_i - D_{i+1}$ a random variable with 0-1 values. Then, by lemma 3.3.6,

$$Pr[X_i = 1] = Pr[(D_i - D_{i+1}) = 1] \geq Pr[(D_i - D_{i+1}) = 1 |\, D_i > 0] \geq 1/2$$

Thus,

$$E[\sum_{i=0}^{T(\epsilon)-1} X_i] \geq T(\epsilon)/2$$

$$Pr[D_{T(\epsilon)} > 0] \leq Pr[\sum_{i=0}^{T(\epsilon)-1} X_i < D] = Pr[\sum_{i=0}^{T(\epsilon)-1} X_i < (1 - (1 - \frac{2D}{T(\epsilon)}))\frac{T(\epsilon)}{2}]$$

We can now deploy the Chernoff bound and the probability will be bounded by (here some calculations are omitted)

$$exp\left[-(1 - \frac{2D}{T(\epsilon)})^2\frac{T(\epsilon)}{4}\right] \leq 2^{-M^2} = \epsilon/n$$

The second part of the lemma now easily follows from the first part, as

$$Pr[T_f \leq 2\lceil\log\Delta\rceil \cdot T(\epsilon)] = Pr[\forall v : T_v \leq 2\lceil\log\Delta\rceil \cdot T(\epsilon)] =$$
$$1 - Pr[\exists v : T_v > 2\lceil\log\Delta\rceil \cdot T(\epsilon)] \geq 1 - \sum_v Pr[T_v > 2\lceil\log\Delta\rceil \cdot T(\epsilon)] >$$
$$1 - \sum_v \epsilon/n = 1 - \epsilon$$ □

By using lemma 3.3.7, the authors proved the main result.

Theorem 3.3.8. *With probability* $\geq 1-2\epsilon$, *all nodes receive message* m *by round* $2\lceil\log\Delta\rceil \cdot T(\epsilon)$. *Furthermore, with probability* $\geq 1-2\epsilon$, *all nodes terminate the execution of* BROADCAST *by round* $2\lceil\log\Delta\rceil \cdot (T(\epsilon) + \lceil\log(N/\epsilon)\rceil)$

We can simplify the complex expression $\lceil\log\Delta\rceil \cdot (T(\epsilon) + \lceil\log(N/\epsilon)\rceil)$ to deduce that algorithm BROADCAST performs broadcasting with high probability and time complexity $\mathcal{O}(D\log n + \log^2 n)$. This algorithm may not be optimal, however it is quite simple and computationally inexpensive. It demands no topological knowledge of the network (a node is not necessary to know its own label), but the nodes should know a priori an upper bound on the size of the network N, as well as the parameter Δ. Furthermore, the algorithm can be deployed without any changes to both directed and undirected radio networks.

The gap between the lower and the upper bound for randomized broadcasting was closed by Czumaj and Rytter [25] and independently by Kowalski and Pelc [53].

Czumaj and Rytter used *selecting sequences*, which in randomized algorithms determine the probability with which all *active* nodes (i.e. nodes that have received the message) will transmit the message at a given round. By carefully designing the selecting sequences, the authors constructed a randomized algorithm that performs broadcasting in $\mathcal{O}(n)$ running time with high probability and another randomized algorithm that completes broadcasting in $\mathcal{O}(D\log(n/D) + \log^2 n)$ with high probability in networks with eccentricity D.

Both algorithms are very similar and their only difference is the choice of the selecting sequence. For the linear time broadcasting, we define the sequence α, where

$$\alpha_k = \begin{cases} 2^{-(k+1)} & 1 \leq k \leq \lceil\log\log n\rceil, \\ \frac{1}{2\log n} & \lceil\log\log n\rceil < k \leq \log n, \\ 1 - \sum_{i=1}^{\log n}\alpha_i & k = 0. \end{cases} \tag{3.5}$$

We can then construct a randomized selecting sequence $\mathcal{J} = \{J_1, J_2, \dots\}$, such that $\mathbf{Pr}[J_i = k] = \alpha_k$. The sequence will be given as input to the algorithm (or it can be constructed by the source node and then transmitted along with the message).

Algorithm 15: Linear-Broadcast (n, $\mathcal{J}$)

```
for $r$ = 1 to $T$ do
    for each active node $v$ do
        node $v$ transmits with probability $2^{-J_r}$
    end
end
```

We can see that the selecting sequence is constructed so that the probability that all nodes in one round transmit with probability 2^{-k} decreases exponentially as k grows, but, for larger values of k, the probability cannot decrease more that a constant. For algorithm Linear-Broadcast, the following theorem can be proved:

Theorem 3.3.9. *For any radio network of size n and $T = \Omega(n)$, algorithm* Linear-Boradcast *completes broadcasting with probability at least $1 - 1/n$.*

The complexity of the algorithm is optimal (as there is a $\Omega(n)$ lower bound for broadcasting when D is not bounded). However, when the eccentricity of the network is bounded, by choosing a different selecting sequence, we can perform broadcasting much faster in time $\mathcal{O}(D\log(n/D) + \log^2 n)$ with high probability. Let us first define $\lambda = \log(n/D)$ and the sequence α' (similar to α),

$$\alpha'_k = \begin{cases} \frac{1}{2\lambda} & 1 \le k \le \lambda \\ \frac{1}{2\lambda} \cdot 2^{-(k-\lambda)} & \lambda < k \le \lambda + \lceil \log\log n \rceil,\ k \le \log n, \\ \frac{1}{2\lambda} \cdot \frac{1}{\log n} & \lambda + \lceil \log\log n \rceil < k \le \log n, \\ 1 - \sum_{i=1}^{\log n} \alpha_i & k = 0. \end{cases} \tag{3.6}$$

We can see that the above distribution does not only depend on the size n, but also on the eccentricity of the network D. The sequences α and α are only different when $k \le \lambda$, i.e. for small enough values of k. Now, based on the sequence α', we can define the randomized sequence $\mathcal{J}^D = \{J_1^D, J_2^D, \dots\}$, such that $\mathbf{Pr}[J_i^D = k] = \alpha'_i$. The algorithm Optimal-Broadcast is quite similar to the algorithm Linear-Broadcast.

Theorem 3.3.10. *For any radio network of size n, eccentricity $D = \Omega(\log^3 n)$ and $T = \Omega(D\log(n/D))$, the algorithm* Optimal-Broadcast *performs broadcasting with probability at least $1 - 1/n$.*

The proof of the running times for algorithms Linear-Broadcast and Optimal-Broadcast follows the same logic. The key idea behind the analysis is to consider a

Algorithm 16: Optimal-Broadcast (n, $\mathcal{J}^D$)

```
for $r$ = 1 to $T$ do
    for each active node $v$ do
        node $v$ transmits with probability $2^{-J_r^D}$
    end
end
```

single path from the source node to an arbitrary node of the network and prove that the end node will have received the message with high probability after the expected time. Let $P = \{v_0, v_1, \dots, v_l\}$ be such a path. We then consider the *layers* of the path P, where each layer $L_P(i)$ is the set of all the in-neighbors of v_i which are not in-neighbors for any node $v_j : j > i$. For the algorithm Linear-Broadcast, direct analysis of the time when the message reaches each layer can lead us to the bound.

As for the algorithm Optimal-Broadcast, we have to analyze separately *small* layers (layers where $|L_P(i)| \leq n/D$) and *large* layers (where $|L_P(i)| > n/D$). We can then show that for each small layer the expected number of rounds needed is $\mathcal{O}(\log(n/D))$, and since there are D layers maximum, the total number of rounds can be shown to be $\mathcal{O}(D\log(n/D))$. As far as large layers are concerned, it can be proved that for each large layer $L_P(i)$ we need expected time $\mathcal{O}(|L_P(i)| \cdot (D/n) \cdot \log(n/D))$. Thus, for all large layers we need $\sum_{i=1}^{l} \mathcal{O}(|L_P(i)| \cdot (D/n) \cdot \log(n/D)) = \mathcal{O}(D\log(n/D))$ expected number of rounds with high probability. By combining the above results, the expected running time can be easily deduced.

Kowalski and Pelc [53] provided an optimal randomized algorithm for broadcasting right after Czumaj and Rytter. The idea behind their algorithm is quite different compared to the algorithm of the latter. We assume that the nodes have labels from the set $\{0, 1, \dots, r\}$, where r is linear to n. We also assume that the nodes know only their own label and the parameter r.

The algorithm works in $\mathcal{O}(D)$ stages and each stage consists of two phases. The first phase consists of $\log(r/D)$ rounds, where in round i of each stage each node transmits with probability $1/2^i$. This ensures that every node with at most r/D active neighbors will receive the message with high probability. The nodes with more than r/D active neighbors are more problematic to deal with and have to been taken care of in the second phase. The second phase consists of only one round, where the probability with each the nodes transmit are carefully chosen so that after k of the neighboring nodes are active ($k > r/D$), the node will receive the message with high probability after $\mathcal{O}(r \log r/k)$ stages. The construction of the probabilities for the second phase is quite complex (see [54] for more details). Moreover, the complicated part of the algorithm analysis is to prove that the algorithm needs only $\mathcal{O}(D)$ stages to complete the broadcasting to the nodes with more than r/D active neighbors.

The novelty of the algorithm lays to the last step of each stage, which allows us to reduce the rounds involved in each stage. This last step ensures that, while the number of rounds is reduced, the remaining uninformed nodes will eventually be informed with high probability and thus broadcasting will be completed. The above algorithm cannot be used straightforward, as it assumes that D is known and that r and D are powers of 2. As a result, the complete algorithm occurs by using the *doubling technique*, i.e. we test different values for D by making exponential jumps. Another important point is that the algorithm can be deployed to directed networks as well.

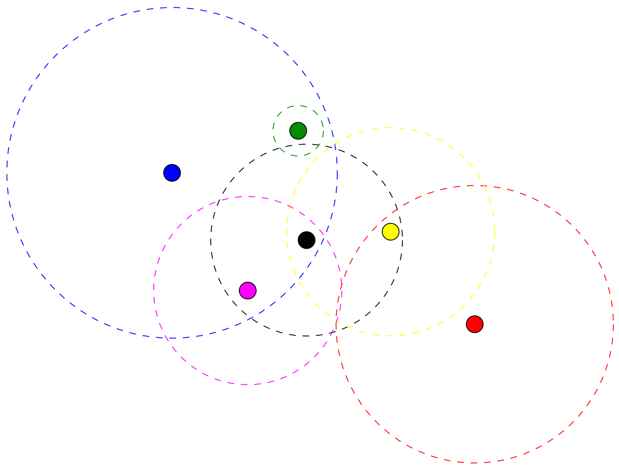
3.4 Broadcasting in Geometric Radio Networks

The classical model of radio networks assumes that the *reachability graph* can be arbitrary, in the sense that any configuration of edges is allowed. In this section, we will deal with a more natural model of radio networks, where we assume that the nodes are placed at points of the euclidean plane and each node is represented by its coordinates. The underlying graph connecting the nodes is no more arbitrary but instead the transmission range of a node v is a region $R(v)$ around node v. A node u can receive messages from v if and only if it belongs to the region $R(v)$. Clearly, the transmission region depends on the power of the node-transmitter, as well as on topographic characteristics of the surrounding region.

Nevertheless, allowing any transmission region is a very general model. We restrict our attention to the case when nodes that can be reached from some node v are nodes belonging to a *disk* of radius r centered at v, and the parameter r, which is called the *range* of v, depends on the power of the transmitter located at v. The reachability graph that corresponds to such radio networks is called a *geometric radio network (GRN).* A node v with range r is connected to some node u at this graph if and only if their euclidean distance is at most r. An example of a GRN can be seen in figure 3.18. We can further restrict our model even assume that all disks have radius 1. The resulting reachability graph is called a *unit disk graph (UDG).*

Since such models of radio networks are a restricted version of the classical model of radio networks, our goal is to construct more efficient broadcasting algorithms which focus on the geometric properties of the models we study. However, the problem of finding the shortest broadcasting scheme (which is NP-hard for an arbitrary radio network) remains NP-hard even when restricted to geometric radio networks. This was proved by Sen and Huson [71]. The intuition behind their reduction was to state the optimal broadcast schedule construction problem in terms of coloring the nodes of a geometric graph. In the same paper, the authors also proposed an algorithm with running time $\mathcal{O}(n \log n)$ that constructs an optimal broadcast schedule when the nodes are situated on a straight line. Broadcasting in networks where the nodes are placed randomly on a straight line was also studied in [68].

In [27], Diks et al. considered broadcasting with restricted amount of knowledge. They assumed that each node knows only its position, its transmission range and the maximum


(a) The nodes on the euclidean plane along with their transmission ranges.

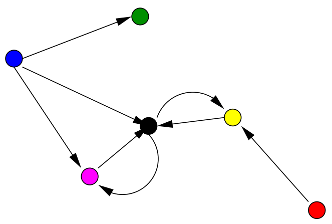

(b) The reachability graph.

Figure 3.18: A geometric radio network (GRN).

transmission range R. The scenario they studied was when nodes are situated at integer points on the line. They first proved a lower bound of $\Omega(D + \frac{\log^2 R}{\log\log R})$ on the broadcasting time of any deterministic protocol. They then provided two algorithms which completed broadcasting in time $\mathcal{O}(D + \log^2 R)$ and $\mathcal{O}(D\frac{\log^2 R}{\log\log R})$. Thus, the gap between the upper and the lower bound exists only when $D = \mathcal{O}(\log^2 R)$. An interesting point here is that the broadcasting time seems to grow when the largest transmission grows, a fact that seems paradoxical. However, as the authors also note, the growth of R likely leads to the decrease of D. Finally, the authors studied the case when the nodes do not know their own transmission range, where they showed that any broadcasting protocol must use time $\Omega(R)$ for a family of networks with $D = 2$.

All the above approaches deal with special families of geometric radio networks. We will now deal with deterministic distributed broadcasting in *arbitrary* geometric radio networks. An interesting approach was proposed in [26] by Dessmark and Pelc, where the authors tried to study how the knowledge of the surrounding area of a node affects the efficiency of broadcasting. They modeled the notion of the restricted knowledge of topology by introducing the *knowledge radius*. When we assume a knowledge radius s, we mean that every node of the network knows the position, range and label of all nodes which are at distance at most s from its own. Their model also assumes that the set of available ranges is known to all nodes and that spontaneous transmissions are allowed.

First, the authors considered the case when the knowledge radius is large enough to exists the largest of all ranges or when it exceeds the maximum distance between any two nodes. Notice that in this case the model is closer to centralized broadcasting. The authors proved that we can perform broadcasting in time $\mathcal{O}(D)$, which is clearly optimal. Then, they examined the opposite case when $s = 0$, i.e. each node knows its label, position and range. They showed that we can construct an algorithm that broadcasts in time $\mathcal{O}(n)$. Let us point here that an algorithm that broadcasts in time $\mathcal{O}(n)$ was also proposed in arbitrary graphs ([13]), but the algorithm only works for symmetric networks.

Another interesting point the authors proved is that distributed broadcasting is more efficient on geometric graphs not because the topology is more restricted, but because the nodes know something additional to their own labels and range: their position on the plane. The latter parameter is used extensively by broadcasting algorithms on geometric radio networks. They particularly showed that there exists a family of geometric graphs where broadcasting needs time $\Omega(n \log n)$ when the nodes do not know their own position.

Finally, the authors showed that when restricted to symmetric geometric networks, we can broadcast even faster. Specifically, assuming the presence of *collision detection* and knowledge radius $s = 0$, they managed to construct an algorithm that broadcasts in time $\mathcal{O}(D + \log n)$ and furthermore showed that the time is asymptotically optimal. The result holds even with the absence of collision detection, however s must be positive.

In [39], Gasieniec et al. tried a different approach to the problem of broadcasting in geometric radio networks. They examined a simple model where each node of the GRN has range 0 or 1 and then a model where the available ranges belong to $[r_{min}, \ldots, r_{max}]$. However, their model assumes complete knowledge of the network topology.

An interesting question here is to study whether distributed deterministic broadcasting can be performed more efficiently when we restrict our attention to UDG networks, i.e. geometric graphs where all nodes have range equal to 1. This question was examined thoroughly in [34] and later in [35]. The authors constructed algorithms for broadcasting with time complexity that depends not only on the eccentricity D, but also on the *granularity* of the network g. The granularity g is the inverse of the minimum euclidean distance between any two nodes of the network. Thus, we can view the granularity as a parameter that measures the *spacing* of the graph.

The authors assumed the classical distributed model where each node knows only its position and the parameter g and no spontaneous transmissions are allowed. In [34], they provided an algorithm that completes broadcasting in time $\mathcal{O}(Dg)$. A matching lower bound was then presented in [35]. One may attempt to compare this result to the $\mathcal{O}(n)$ algorithm for general geometric networks. In this context, it should be noted that the total number of nodes in a network of eccentricity D and granularity g may be at most $\Omega(D^2 g^2)$ or as small as $\mathcal{D}$. Hence, the algorithm of [26] is generally slower than the $\mathcal{O}(Dg)$ algorithm.

In [34], the authors also studied the same model, but this time with *spontaneous wake-up*. They provided two algorithms for broadcasting, the one working in time $\mathcal{O}(D + \log^2 g)$ and the other in time $\mathcal{O}(D \log g)$. The choice of the faster algorithm depends on the values of the parameters D and g. Thus, broadcasting can be completed in time $\mathcal{O}(\min\{D \log g, D + \log^2 g\})$. Moreover, the authors showed that this time is asymptotically optimal. Clearly, this establishes a gap between broadcasting with or without spontaneous wake-up.

Finally, in [33] Elsässer et al. followed a totally different direction on studying geometric radio networks. They considered broadcasting in *random geometric graphs*, where n nodes are placed uniformly at random in $[0, \sqrt{n}]^2$ and the transmission radii of the nodes vary according to a power law distribution. In particular, a node is assigned a transmission radius larger than some value r with probability proportional to $r^{1-\alpha}$, where $\alpha \in (1, 3)$ is some fixed constant. The authors provided algorithms for two variations of the probability distribution function and showed that radio networks with low average transmission radius can be designed, where broadcasting is performed exponentially faster than in the case when all nodes have equal transmission ranges.

3.5 Fault-Tolerant Broadcasting

We have studied so far radio networks under the assumption that no failures occur during the execution of an algorithm. Nevertheless, due to the growth of the network size, networks become more vulnerable to component failures. These failures may be either *link* or even *node* failures. Thus, designing algorithms that take into consideration the possibility of failures and accomplish the communication task (in our case broadcasting) even when failures occur is becoming more and more important. We call such algorithms *fault-tolerant* algorithms. Pelc in [65] presents a detailed survey on fault-tolerant algorithms designed for broadcasting in communication networks. Furthermore, for a more detailed presentation of fault-tolerant protocols on communication networks, see [60].

There have been many different approaches to how we model a *failure* in a network. In the setting of the radio networks, we are interested only in node failures. However, we have two different types of node failures:

Crash Faults : The faulty node does not receive or transmit any messages. However, faulty nodes are not allowed to alter transmitted messages.

Byzantine Faults : The model where we consider a *Byzantine fault* is a worst-case model. Faulty components behave arbitrarily, i.e. they not only stop transmitting or receiving, but they can also alter the context of a message or even the broadcasting schedule. Clearly, such a behavior is not very common; nevertheless, it can model the case of someone trying to destroy the communication process. Furthermore, since Byzantine faults represent a worst-case scenario, algorithms that work well in this context work also well under any fault scenario.

Another crucial characteristic of a network with possible faults is the distribution of the faults. It is natural to impose a limit on the number of faults that may occur, since otherwise communication may be impossible. We will examine two fault models, along with some small modifications.

The Bounded Model : We assume an upper bound on the number of nodes of any neighborhood that may fail. The source is always fault-free. The goal is to ensure that the message reaches all fault-free nodes.

The Probabilistic Model : We assume that faults occur randomly and independently of each other, with some specified probability distribution. Since we adopt this scenario, the best we can guarantee is broadcasting with high probability. Thus, we want to construct *almost-safe* algorithms, where broadcasting completes successfully with probability at least $1 - 1/n$, for sufficiently large n.

The first to study fault-tolerant protocols in radio networks were Pagani and Rossi [64]. The authors considered only *transient* faults, i.e. non-permanent faults. Moreover, the fault-tolerant protocol they presented tolerates even disconnections and network partitions, with the assumption that they are eventually repaired. This model is more suitable for the description of mobility or communication failure over wireless links.

Kranakis, Krizanc and Pelc [57] first examined broadcasting with *permanent* node failures, i.e. the fault status of a node does not change during broadcasting. Furthermore, they assumed that the location of the faulty nodes is unknown, and thus we consider the worst-case scenario. One more thing we should take into consideration is that the configuration of faults may be such that disconnects the network. In this case, broadcasting is completed when the message reaches all fault-free nodes of the connected component containing the source.

The authors presented fault-tolerant protocols for two configurations of the network graph, where the nodes are situated at integer points on a line or at integer grid points of a square (or hexagonal) mesh and have equal transmission ranges. They also distinguished the protocols to *adaptive* and *non-adaptive*. In the case of non-adaptive algorithms, in the presence of at most t faults, they showed that fault-tolerant broadcasting completes in time $\mathcal{O}(D + t)$, while showing a lower bound of $\Omega(D + t)$. Thus, we have an optimal algorithm for non-adaptive fault-tolerant broadcasting. In the case of adaptive algorithms, they presented an optimal algorithm that works in time $\mathcal{O}(D + \log(\min\{R, t\}))$, where R is the transmission range of the nodes.

Let us now turn our attention to Byzantine faults. The first analysis of reliable broadcasting in radio networks for the case of Byzantine adversaries was provided by Koo [50]. The author examined protocols only for the case where the nodes are situated on a square grid and have equal transmission ranges r. However, since a Byzantine adversary can behave in an arbitrary manner, there is no restriction on the behavior and thus a corrupted node may keep sending noise in order to prevent any communication within its radius. In order to avoid this scenario, we have to assume some restriction to the communication process: we fix a schedule where the nodes send messages in turns. Consequently, a corrupted node may transmit only when its turn comes. More over, a faulty node cannot spoof the address of some fault-free node. Finally, instead of bounding the number of faulty nodes overall the network, the author considered a bound on the number of faulty nodes within the range of a fault-free node (we denote this by t).

Under the above assumptions, Koo proposed a simple algorithm that performs reliable broadcasting when there are at most $t = \frac{1}{4}r(r + \sqrt{r/2} + 1) - 3$ (roughly a $1/4\pi$ fraction) faulty nodes within the radius of any fault-free node. The basic idea of the protocol is that nodes wait until they hear the same message from $t + 1$ neighbors before they accept it as

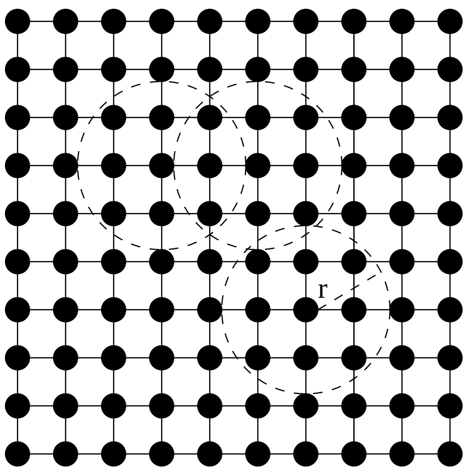


Figure 3.19: Nodes placed on a square grid. Each node has transmission range r.

the correct message. Then, they broadcast the correct message just once. This protocol guarantees that no fault-free node ever accepts a wrong message. On the other hand, we can also prove that secure broadcasting is impossible when $t \geq \lceil \frac{1}{2}r(2r+1)\rceil$ (roughly a $1/\pi$ fraction). Thus, a small gap between the achievability and impossibility bound still remains. Both bounds hold when we consider an L_2 metric, however, similar results are obtained for the L_1 and L_∞ metrics.

Bhandari and Vaidya [5] closed the gap for the case of the L_∞ metric and showed that secure broadcasting is possible if and only if $t < \frac{1}{2}r(2r+1)$. Furthermore, they also studied reliable protocols assuming crash failures instead of Byzantine failures.

In [51], the authors removed some of the assumptions made in [50] and [5] and allowed each faulty node to cause a (known) bounded number of collisions and spoof the addresses of other nodes in the network. In this harder model, they showed that the maximum tolerable t (number of faulty nodes within a radius of a faulty-free node) is equal to the maximum tolerable t when collisions and address spoofing are not allowed. As a result, by causing collisions and spoofing addresses an adversary may be able to decrease the efficiency of broadcasting, but the task remains feasible.

Finally, Pelc and Peleg [66] studied fault-tolerant broadcasting in radio networks under a probabilistic failure model. At each step, the transmitter of each node may fail with fixed constant probability $p < 1$, and failures are independent. They authors showed that, assuming *node-omission* failures, *almost-safe* broadcasting is feasible for any $p < 1$. For the case of *malicious* failures, almost-safe broadcasting is possible if and only if $p < (1-p)^{\Delta+1}$. As far as the time complexity of broadcasting is concerned, we can prove that almost-safe broadcasting in time $\mathcal{O}(opt + \log n)$ is impossible for some graphs, whereas we can construct an almost-safe protocol with execution time $\mathcal{O}(opt \cdot \log n)$ for any graph and for both types of failures.

Chapter 4

Gossiping

Gossiping consists a basic communication task in modern communication networks. Gossiping corresponds to the everyday task where each individual holds a distinct piece of information and wants to inform all the other individuals about it. In the setting of a radio network, we assume that each node of the network possesses a unique message and the goal is to distribute the message of every node to all the nodes in the network. Clearly, in order to ensure the feasibility of gossiping, the underlying graph must be *strongly connected*, i.e for any pair of nodes u, v, there exists a path from u to v. Gossiping is also commonly referred as *all-to-all communication.*

The study of gossiping in radio networks began much later compared to the task of broadcasting. The same variants of the communication model we examined for broadcasting are also considered for gossiping. Consequently, we distinguish the gossiping protocols to *randomized* or *deterministic*, *centralized* or *distributed*, *oblivious* or *adaptive.* One more distinction we find in the literature and concerns only gossiping is whether we assume an upper bound on the size of the messages transmitted. This distinction is not necessary for broadcasting, since we have only one message that needs to be disseminated to all the nodes of the network. In gossiping, however, it would be more efficient if we allowed the possibility to combine many messages collected from a node in a single message, which we transmit in a single round.

Before we move to the description of gossiping algorithms in the several communication settings, let us note that, whereas gossiping at first glance seems to be more difficult than broadcasting, there are many communication scenarios (such as distributed deterministic gossiping) where we have no clue whether this fact holds. Non-trivial lower bounds for gossiping have been proved only in the case of bounded-size messages.

4.1 Deterministic Distributed Gossiping

In this section, our main focus will be the study of gossiping in distributed radio networks. The model was thoroughly presented in the previous chapter, so let us just remind that we assume that the nodes are only aware of their own label and possibly the size n, the maximum eccentricity D, or the maximum in-degree Δ of the network.

The main research direction in deterministic gossiping assumes that the messages transmitted can be of arbitrary size. That means that any gossiping protocol can encapsulate many messages in one message and transmit them in just one round. A trivial algorithm that performs gossiping is the classical ROUND-ROBIN-based algorithm which was used for the broadcasting task and can be slightly modified in order to perform gossiping as well. Thus, we easily obtain a trivial upper bound of $\mathcal{O}(n^2)$ time for deterministic gossiping in ad-hoc radio networks.

The first to propose a non-trivial algorithm for gossiping in ad hoc radio networks were Chrobak, Gasieniec and Rytter [18]. Their algorithm was the first subquadratic gossiping algorithm, having running time $\mathcal{O}(n^{3/2}\log^2 n)$. Another interesting approach to distributed gossiping was by Clementi et al. [23], where the authors proposed a gossiping algorithm running in time $\mathcal{O}(D\Delta^2\log^3 n)$. Gasieniec and Lingas [40] provided another algorithm with running time $\mathcal{O}(nD^{1/2}\log^3 n)$. The latter algorithm performs gossiping faster than the algorithm from [18] for networks of maximum eccentricity $D = \mathcal{O}(n^a), a < 1$. The last results imply that graphs with large maximum eccentricity D consist a bottleneck for deterministic distributed gossiping. Xu in [73] fine-tuned the algorithm from [40] to produce a gossiping algorithm with running time $\mathcal{O}(n^{3/2})$.

We should also mention the work of Ravishankar and Singh, who presented distributed gossiping algorithms for networks with nodes placed randomly on a line [69] and on a ring [70]. Another interesting line of research is to apply a limit on the size of messages transmitted ([3],[22],[17]). Last, Chlebus et al. [14] studied *oblivious* algorithms for gossiping in ad-hoc radio networks and showed upper and lower bounds for the time needed for gossiping.

4.1.1 A subquadratic algorithm for gossiping

Chrobak et al. [18] were the first to construct a subquadratic algorithm for gossiping. The algorithm DOGOSSIP is based on the observation that gossiping is not just the composition of$n$ simultaneous broadcasts. Instead, a node may wait until it collects more than one message and encapsulate the messages in a single message before transmitting. Since we use broadcasting as a subroutine for gossiping, we assume from now on that we have an algorithm BROADCAST which completes broadcasting in time $B(n)$.

First, let us describe a basic procedure which we will later deploy. Assume that every node v holds an integer value r_v, where $0 \leq r_v \leq n$. The procedure FINDMAX succeeds in electing a single node v which holds the maximum integer, i.e. all the nodes have to agree about which node holds the largest value. FINDMAX uses binary search in order to specify the node with the largest integer.

Procedure FINDMAX only finds the largest value among the nodes of the network. However, we also want to pinpoint which node holds this largest value. For that, we apply

```
Procedure FindMax
a = 0 ;
b = n ;
repeat
    c = ⌈(a + b)/2⌉ ;
    Foreach node v such that $r_v \in [c, b]$ run Broadcast sending the message
    $[c, b]$ ;
    wait $B(n)$ rounds ;
    if the message is received then
        a = c
    else
        b = c − 1
    end
until a = b ;
return a ;
```

again binary search, but this time we set the values as follows: $r'_v = v$ if node v holds r_{max}, else we set $r'_v = 0$.

We can now present the algorithm DoGossip (algorithm 18). We denote by $S(v)$ the set of all distinct messages that node v holds. Whenever node v transmits, the whole set $S(v)$ is transmitted encapsulated in a single message. Furthermore, when a new message is received, it is added to $S(v)$.

```
Algorithm 18: DoGossip
Run $\sqrt{B(n)} \log n$ of RoundRobin ;
repeat
    Run Find-Max to find the node $v_{max}$ with the largest set $S(v)$ ;
    Use Broadcast to transfer $S(v_{max})$ to all the nodes ;
    For every node $v$, remove from $S(v)$ the messages from $S(v_{max})$ ;
until $\max_v |S(v)| = 0$ ;
return a ;
```

Let us now provide some analysis for the running time of the algorithm DoGossip. The first part of the algorithm needs $\mathcal{O}(n\sqrt{B(n)}\log n)$ time. Let $a = \sqrt{B(n)}\log n/n$ and also assume that at the beginning of any iteration of the main loop we have m distinct messages. The consecutive executions of the RoundRobin procedure will have distributed each message to at least $a \cdot n$ nodes and thus we have at least $a \cdot n \cdot m$ copies of the messages. Consequently, $\max_v |K_v| \geq a \cdot m$. Now, at the end of the iteration, the number of distinct messages will be at most $(1 - a) \cdot m$. Since at the beginning of the algorithm we have n distinct messages, after i iterations, the number of messages left will be $(1 - a)^i \cdot n$. We thus need $\mathcal{O}(1/a \cdot \log n) = \mathcal{O}(n/\sqrt{B(n)})$ iterations of the main loop in order to exhaust all messages. Each iteration needs $B(n) \cdot \log n$ rounds and so the second part of the algorithm also has running time $\mathcal{O}(n\sqrt{B(n)}\log n)$. The theorem follows.

Theorem 4.1.1. *Algorithm* DoGossip *completes gossiping in time $\mathcal{O}(n\sqrt{B(n)}\log n)$, where $B(n)$ is the running time of the broadcasting algorithm used.*

If we use the broadcasting algorithm of [18], which completes broadcasting in time $\mathcal{O}(n\log^2 n)$, we come up with a time complexity $\mathcal{O}(n^{3/2}\log^2 n)$. There has been proposed an even faster broadcasting algorithm in [61], however the running time remains $\tilde{\mathcal{O}}(n^{3/2})$.

4.1.2 A faster gossiping algorithm

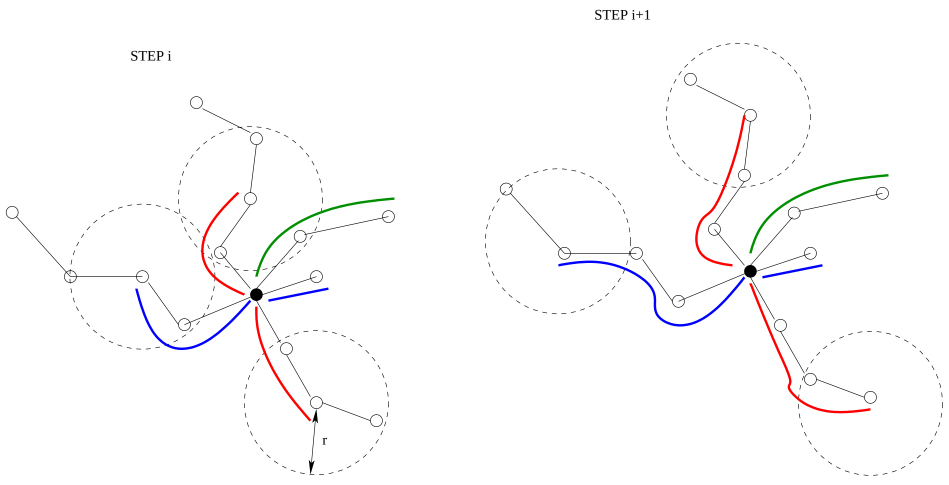
Algorithm DoGossip does not collect any information about the topology of the network in order to increase the efficiency of gossiping. We present here an algorithm proposed by Gasieniec and Lingas [40], which is *strongly adaptive* and exploits information about the neighborhood of nodes in order to gather more efficiently messages from the nodes of the network.

The algorithm initially assumes that every node knows the maximum eccentricity of the network D. However, we can drop this assumption by using the standard *doubling technique* we used for broadcasting, without altering the asymptotic running time of the algorithm. The detailed algorithm CollateAndBroadcast is presented below (algorithm 19).

Algorithm 19: CollateAndBroadcast(r)

```
Run RoundRobin $r$ times ;
Choose a node $\lambda$ by using the procedure FindMax ;
for $i = 1, \dots, \lceil D/r \rceil$ do
    (a) Store information gathered from node $\lambda$ in a graph representation
    $G_{ir}(\lambda)$ covering all nodes within in-radius $ir$ from $\lambda$. $G_{ir}(\lambda)$ is a union of
    paths directed from the nodes to node $\lambda$, which we construct to start from
    different start points. Root the tree $G_{ir}(\lambda)$ from node $\lambda$ and fix a
    post-order for the nodes of $G_{ir}(\lambda)$ ;
    (b) Use Broadcast to transfer the information from step (a) to all nodes
    in the network ;
    (c) Every node $v$ in $G_{ir}(\lambda)$ transmits according to the post-order its
    neighborhood of radius $r$ $N_v(r)$, as well as the information received so far.
    Thus, node $\lambda$ collects information from the $(i+1)r$-neighborhood of $\lambda$ ;
end
```

Figure 4.1 depicts schematically the expansion of the graph representation G_{ir} after the execution of one iteration. Let us analyze the time complexity of algorithm Collate-AndBroadcast. First, we set parameter r equal to $\lceil\sqrt{D}\rceil$. Clearly, the execution of the



(a) The graph representation G_{ir} after step i

(b) The graph representation $G_{(i+1)r}$ after step $i+1$

Figure 4.1: Expansion of the graph representation during the execution of algorithm CollateAndBroadcast

RoundRobin procedure r times needs $\mathcal{O}(\sqrt{D} \cdot n)$ rounds. The selection of the central node λ takes $\mathcal{O}(B(n) \log n)$ time, where $B(n)$ is the running time of Broadcast. The last stage of the algorithm is executed $\mathcal{O}(\sqrt{D})$ times. Since step (c) can be executed in linear time to n, the most expensive step of the last stage is the execution of Broadcast. Thus, the last stage lasts $\mathcal{O}(\sqrt{D} \cdot B(n))$ rounds. By summing up the running times of each stage, we conclude that the total running time of the algorithm CollateAndBroadcast is $\mathcal{O}(\sqrt{D} \cdot B(n))$. In view of the fact that broadcasting can be performed in $\mathcal{O}(n \log^2 n)$ time, we have proved the following theorem holds.

Theorem 4.1.2. *The algorithm* CollateAndBroadcast *performs gossiping in any radio network of size n and maximum eccentricity D in $\tilde{\mathcal{O}}(\sqrt{D} \cdot n)$ time.*

Notice that by setting $r = \sqrt{D} \cdot \log n$, we would achieve an even better time complexity of $\mathcal{O}(\sqrt{D} n \log n)$. However, we would have to make the assumption that the nodes know the size of the network n, an assumption which would like to avoid.

Clearly, CollateAndBroadcast outperforms DoGossip when $D = \mathcal{O}(n^a)$, for any constant $a < 1$. We should state here that the authors also provided an alternative version of the algorithm which performs gossiping in time $\mathcal{O}(D\Delta^{3/2} \log^3 n)$.

Xu [73] proposed an improvement over algorithm CollateAndBroadcast and managed to drop the running time of gossiping to $\mathcal{O}(n^{3/2})$. The idea of Xu was to collect information not only about the in-paths of the central node, but also from the out-paths. This way, we can perform the broadcasting function of step (b) in algorithm CollateAndBroadcast in just linear time, without using the procedure Broadcast. Specifically, we will construct a graph representation of the out-neighborhood of the graph similar to graph G_{ir} which we call G_{ir}^{+}. G_{ir}^{+} is a directed tree, where all edges are directed from the central node to the leaves, and spans over all nodes within an in-radius of ir. The information about the topology of G_{ir}^{+} will be distributed to all nodes along with G_{ir}. However, in order for each node to know a path from the central node λ to itself, we have to run procedure BROADCAST before the initiation of the algorithm. The rest of the algorithm remains the same as algorithm COLLATEANDBROADCAST and thus the analysis that follows is quite similar. Consequently, we obtain the following theorem.

Theorem 4.1.3. *Gossiping in arbitrary graphs of size n and maximum eccentricity D can be completed in time $\mathcal{O}(B(n)+\sqrt{D}\cdot n)$, where $B(n)$ is the time complexity of broadcasting.*

Since the eccentricity D of the graph is at most n, theorem implies that gossiping can be performed in time $\mathcal{O}(n^{3/2})$. The algorithm of Xu is the currently fastest deterministic gossiping algorithm for ad-hoc radio networks.

4.1.3 Gossiping with Bounded Messages

In this section we will briefly present gossiping under the assumption that we impose an upper bound on the size of messages transmitted. First, Bar-Yehuda et al. [3] considered gossiping in radio networks where the size of the messages is limited to $\mathcal{O}(\log n)$ bits. They proposed a randomized distributed algorithm that performs *multi-broadcasting* (broadcasting of $k > 1$ distinct messages originated from k nodes) in time $\mathcal{O}((D+k)\log\Delta\log n)$. Thus, we obtain an $\mathcal{O}(n\log n\log\Delta)$ randomized gossiping algorithm.

Clementi et al. [22] also examined gossiping when we assume messages of size at most $\mathcal{O}(\log n)$. They provided lower bounds for both randomized and deterministic distributed gossiping under this assumption. Furthermore, they proposed a gossiping algorithm that is based on *strongly selective families* and runs in time $\mathcal{O}(n\Delta^2\log^3 n)$.

Christersson, Gasieniec and Lingas [17] assumed a more general bound on the message size. Specifically, they express this upper bound in terms of an integer function $b(n)$, where n is the number of nodes in the network. That means that we can transmit a combined message that contains at most $b(n)$ unit messages. We call this restriction of the gossiping problem *$b(n)$-gossiping.*

The authors initially study *directed* networks and show that we can perform deterministically $\sqrt{n}$-gossiping in an arbitrary graph in time $\tilde{\mathcal{O}}(n^{3/2})$. The algorithm ($\sqrt{n}$-GOSSIP) is quite similar to the algorithm DOGOSSIP from [18]. Next, they show how to transform $\sqrt{n}$-GOSSIP to an algorithm that performs $b(n)$-gossiping for $0 \leq b(n) \leq \sqrt{n}$ and runs in $\tilde{\mathcal{O}}(n^2/b(n))$ time. These upper bounds are tight up to a polylogarithmic factor, since we can easily prove that $b(n)$-gossiping needs at least $\Omega(n^2/b(n))$ rounds. This means that, in order to achieve gossiping faster than $\tilde{\mathcal{O}}(n^{3/2})$, the size of the combined messages must exceed $\sqrt{n}$.

As far as gossiping in undirected graphs is concerned, the authors proposed faster algorithms. Specifically, they proved that even 1-gossiping can be done deterministically in time $\tilde{\mathcal{O}}(n^{3/2})$. If we use randomization, the time for 1-gossiping is decreased to $\tilde{\mathcal{O}}(n)$. Finally, they showed that $\mathcal{O}(n^t)$-gossiping can be performed in $\mathcal{O}(n^{2-t})$ rounds.

4.2 Gossiping with Polynomially Large Labels

The algorithms presented in the above section assume that the node labels are integers and the largest value is linear to the size of the network n. Equivalently, each node chooses a label from the set $\{1, 2, \ldots, c \cdot n\}$ for some constant c. A very interesting model of radio networks occurs when we consider polynomially large node labels. Our goal is to construct gossiping algorithms with time complexity that depends only on the size n and not on the largest label N (however, it is acceptable if N is used in a logarithm). The gossiping algorithms presented so far can not be modified so as to satisfy this property. Thus, we have to introduce new gossiping algorithms, which manage to avoid the expensive N factor. Gossiping in this context was first studied by Gasieniec, Pagourtzis and Potapov [41], where gossiping was performed in $\tilde{\mathcal{O}}(n^{5/3})$ time. In the journal version [42], the running complexity was improved to $\tilde{\mathcal{O}}(n^{3/2})$. The faster known gossiping algorithm was proposed by Gasieniec, Radzik and Xin [45] and has running time $\tilde{\mathcal{O}}(n^{4/3})$.

Let us first introduce some terms necessary in order to describe the gossiping algorithms of this section. We define a message to be *secure* if it has been transfered to all the nodes of the network, otherwise we say that the message is *insecure*. A *dormant* node is a node whose original message is secure. An *active* node is a node which is not dormant. We also define an *active path* to be a simple path such that all nodes of the path but the last one are active.

Both of the algorithms of this section use the principle of *quasi-gossiping*. A procedure that completes *quasi-gossiping* guarantees that, upon its completion, every active message has been communicated to at least one dormant node. Full gossiping can be achieved if we execute a quasi-gossiping procedure and then repeat the transmissions in exactly the same order they occurred during quasi-gossiping.

We will first describe the gossiping algorithm GossipLL by Gasieniec et al. [42]. The algorithm uses for broadcasting the algorithm DoGossip from [18], which completes broadcasting in time $\mathcal{O}(n \log n \log N)$. Let us denote this algorithm by Broadcast. The algorithm consists of four stages and heavily depends on the use of different kinds of *selectors*. We should also note that the use of RoundRobin is avoided, as the factor N will then appear in the time complexity. The optimal value of parameter k will be determined later.

Let us provide some analysis and insight on the algorithm GossipLL. The purpose of stage 1 is to ensure that, by completion, each node of the network has fewer than k active neighbors. Since the most expensive part of stage 1 is the selection of a central node, which costs $\mathcal{O}(n \log^2 N \log n)$, (the linear selective family is of size $\mathcal{O}(n \log N)$) and the iterations will be at most $2n/k$, we conclude that the running time of stage 1 is $\mathcal{O}((n^2 \log^2 N \log n)/k)$.

Algorithm 20: GossipLL

Stage 1: We execute an iterative procedure. At first, all the active nodes run a *linearly k-selective family*. Due to the properties of such a family, if a node has at least k active neighbors, it will collect at least $k/2$ insecure messages. We then choose a node λ among the nodes with at least $k/2$ insecure messages. If no such node exists, stage 1 terminates. Node λ starts Broadcast to distribute all its messages to all the nodes of the network.

Stage 2: At the beginning of stage 2, we run a *strongly k-selective family k* times. We then repeatedly choose a central node λ with at least k insecure messages and run Broadcast with λ as the source. The iterations stop when every node has less than k insecure messages.

Stage 3: We run again a *strongly k-selective family k* times.

Stage 4: During this stage, every node repeats exactly the sequence of its transmissions from stages 1 and 2.

The goal of stage 2 is to break long active paths into active paths of length less than k. This way, we reduce the maximum distance from an active node to a dormant node. The second stage runs for $\mathcal{O}(k^3 \log N + (n^2 \log^2 N \log n)/k)$ rounds (a strongly k-selective family has size $\mathcal{O}(k^2 \log N)$).

After stage 3, if there was some dormant node at the beginning of stage 3, each insecure message reaches a dormant node and thus quasi-gossiping succeeds. Else, by the end of stage 3, full gossiping is complete. Clearly, stage 3 runs in time $\mathcal{O}(k^3 \log N)$. Now, if at the beginning of stage 4 only quasi-gossiping is complete, the repetition of the transmissions from the first two stages guarantees that all messages will reach all nodes. Thus, the algorithm completes gossiping in time $\mathcal{O}(k^3 \log N + (n^2 \log^2 N \log n)/k)$. The optimal running time is achieved when $k = n^{1/2}(\log N \log n)^{1/4}$ and thus we obtain the following theorem.

Theorem 4.2.1. GossipLL *completes gossiping in an arbitrary radio network with large labels in time $\mathcal{O}(n^{3/2} \log^{7/4} N \log^{3/4} n)$.*

An interesting thing we should note here is that GossipLL can be modified to complete gossiping in undirected radio networks in substantially less time, namely only in $\mathcal{O}(n \log^2 N \log^2 n)$ time.

The gossiping algorithm by Gasieniec, Radzik and Xin [45] is quite similar to GossipLL and depends on the use of a different selector, the *path selector*. The *path selector* has a very important property, which is described by the following lemma.

Lemma 4.2.2. *Consider a directed simple path $P =< v_0, \dots, v_m >$, where the neighborhood of the path is smaller or equal to k. An application of a path selector $S_{N,k}$ succeeds in that all the nodes of the path deliver their messages to the endpoint v_m .*

We will also need a procedure Disperse, which is responsible for the distribution of large enough combined messages to all the nodes of the network. The detailed algorithm is presented below.

Procedure Disperse(k)

```
while a node with $\geq k$ active messages exists do
    select such a node $\lambda$ ;
    run Broadcast with $\lambda$ as source ;
end
```

Algorithm 22: GossipPS

```
(Stage 1):
$q = n$ ;
while $q \geq k$ do
    apply a $(N, q, q/4)$-selector ;
    Disperse($q/4$) ;
    $q = q/2$ ;
end
(Stage 2):
for 1 to $\lceil \log k \rceil + 1$ do
    apply a path selector $\mathcal{S}_{N,k}$ ;
    Disperse($k/2$) ;
end
(Stage 3):
repeat transmissions from stages 1 and 2 in the same order ;
```

Clearly, the analysis is similar to the corresponding analysis of GossipLL. After stage 1, we can prove that every node has less than k active neighbors. We can show that r executions of procedure Disperse(k) need $\mathcal{O}((n/k+r)n\log^3 n)$ rounds, thus stage 1 needs $\mathcal{O}(n^2/k\log^3 n)$ rounds. The second stage reduces the size of neighborhoods of active paths (by neighborhood of a path we mean the union of the neighbors of every node of the path). Specifically, the use of path selectors guarantees that at the beginning of the last iteration of stage 2, the size of the neighborhood of each active path is less than k. The last application of the path selector will either complete quasi-gossiping or even full gossiping. The running time of stage 2 is $\mathcal{O}((n/k+\log k)n\log^3 n)$ for Disperse and $\mathcal{O}(k^2\log^3 n\log k)$ for the path selector (a path selector $\mathcal{S}_{n,k}$ has size $\mathcal{O}(k^2\log^3 n)$). Stage 3 is needed for the case of quasi-gossiping, where the repetition of the transmissions ensures that full gossiping completes. By summing up the time for each stage and by choosing the optimal value for k, we can show that algorithm GossipPS runs in time $\mathcal{O}(n^{4/3}\log^{10/3} n)$.

Theorem 4.2.3. GossipPS *completes gossiping in an arbitrary radio network with large labels in time* $\mathcal{O}(n^{4/3}\log^{10/3} n)$.

We observe that the two algorithms do not differ much, but for the use of a more specialized and more expensive selector. Basically, the authors replaced the repeated use of broadcasting with the use of a modified selector which guaranteed stronger properties for the progress of gossiping. This way, quasi-gossiping performs more efficiently.

4.3 Randomized Gossiping

In this section, we will study algorithms for gossiping in radio networks that use randomization. The first randomized algorithm for gossiping was proposed by Chrobak, Gasieniec and Rytter [19] and completed gossiping in $\mathcal{O}(n\log^4 n)$ time. The running time was improved to $\mathcal{O}(n\log^3 n)$ by Liu and Prabhakaran [59]. The currently fastest known gossiping algorithm in ad hoc radio networks is by Czumaj and Rytter [25] and runs in time $\mathcal{O}(n\log^2 n)$.

Let us first present the gossiping algorithm of Chrobak et al. [19] in full detail. The algorithm is based on two key ideas. The first one is to use a faster broadcasting procedure LtdBroadcast (*limited broadcast*), which succeeds in transmitting a message from a node to at least k nodes of the network, instead of a relatively more expensive full broadcast. Clearly, LtdBroadcast will be used as a basic component for the algorithm. However, we must use a small number of limited broadcasts and we must guarantee that every message will participate in such a broadcasting. In order to achieve these goals, the algorithm implements a randomized procedure, which we call *Distributed Coupon Collection* (DistCouponColl).

LtdBroadcast$_v(k)$ is based on the broadcasting algorithm Selector-Broadcast [18], which has a running time of $\mathcal{O}(n\log^2 n)$. Here, we modify the algorithm in order to perform limited broadcasting in time $\mathcal{O}(k\log^2 n)$. The main difference is that we limit the rounds executed to $\gamma k\log^2 n$ for some constant γ. Furthermore, we may assume that, at the beginning of the procedure, other nodes apart from the source may be active, in which case they behave as if they have already received the message. We can prove the following lemma.

Lemma 4.3.1. *Assuming that initially exactly one node u is active and that the procedure* LtdBroadcast$_v(k)$ *is executed simultaneously by all nodes, after the procedure stops, at least k nodes will receive the message from u.*

Let us now turn our attention to the problem DistCouponColl. We can describe the problem as follows. We think the set of nodes V as a set of bins and the set of messages M as a set of coupons. Denote by M_v the set of messages in node v. Furthermore, assume that each message has at least k copies, i.e. belongs to at least k nodes. The procedure (DCC) is quite simple: at each round, we open each bin with probability $1/n$. We then distinguish two cases:

- **Exactly one bin v is opened**: All messages in M_v are collected.
- **No bin or more that two bins are opened**: We collect no message.

The question that occurs now is how many rounds do we need in order to collect all messages (coupons) with high probability. The following lemma answers this question.

Lemma 4.3.2. *Assume that we have n bins and n coupons and that each coupon has at least k copies, each copy belonging to a different bin. Let δ a constant, where $0 < \delta < 1$. Then after s rounds of DCC ($s = (4n/k)\ln(n/\delta)$), all coupons will be collected with probability $1 - \delta$.*

Proof. Let us assume that $n \geq 2$ (the case $n = 1$ is trivial). Denote by $X_{m,j}$ the fact that coupon m is collected in round j. Then, for any m and k we have:

$$Pr[X_{m,j}] \geq \frac{k}{n}\left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{k}{n}\left(1 - \frac{1}{n}\right)^{n} \geq \frac{k}{4n}.$$

Thus, the probability that message m is not collected in s rounds is:

$$Pr[m \text{ not collected}] \leq n\left(1 - \frac{k}{4n}\right)^{s} \leq ne^{-\frac{sk}{4n}} \leq \delta.$$

□

The algorithm RandGossip is a combination of DistCoupColl and LetBroadcast and is presented below.

Algorithm 23: RandGossip(ϵ)

```
$\delta = \epsilon / \log n$ ;
for $i = 0$ to $\log n - 1$ do
    $s = (4n/2^i)\ln(n/\delta)$ ;
    for $s_i$ times do
        Each node $v$ gets active with probability $1/n$ ;
        Each node $v$ executes LtdBroadcast$_v(2^{i+1})$ ;
    end
end
```

Let us now provide the analysis of the algorithm RandGossip(ϵ). Clearly, the running time can be easily shown to be $\mathcal{O}(n \log^3 n \log(n/\epsilon))$. Consider now the beginning of the first iteration ($i = 0$), where each message is contained in its starting node. The algorithm tries to maintain the invariant that after the ith iteration, each message is in at least 2^{i+1} nodes. If we manage to keep the invariant true until the last iteration, then gossiping will be completed successfully. Thus, we turn our attention to computing the probability that the invariant fails. Since each iteration is basically an execution of procedure DCC s times, the probability that the invariant fails, assuming that it has not failed so far, is at most δ. Summing up, the probability of failure is at most $\log n \cdot \delta = \epsilon$.

Notice that RandGossip is a Monte Carlo algorithm, whereas we would most be interested in a Las Vegas algorithm. To obtain such an algorithm, we modify the algorithm the following way. We first run RandGossip $(1/n)$, which has running time $\mathcal{O}(n \log^4 n)$. If the algorithm fails, we inform all the nodes using some broadcasting procedure and then we execute a simple RoundRobin with running time $\mathcal{O}(n^2)$. Thus, the expected running time of the modified algorithm is $\mathcal{O}(1/n(n^2) + (1 - 1/n)n \log^4 n) = \mathcal{O}(n \log^4 n)$. Consequently, we have proved the following theorem.

Theorem 4.3.3. *There exists a Las Vegas algorithm that completes gossiping in any radio network of size n with expected running time $\mathcal{O}(n \log^4 n)$.*

Algorithm RandGossip was improved by Liu and Prabhakaran [59] to run in time $\mathcal{O}(n \log^3 n)$. The algorithm runs in two phases. The first phase consists of $\log n$ executions of RoundRobin. After this, each message has at least $\log n$ copies in the network, which is necessary for the next phase to have a good success probability. The second phase is exactly like RandGossip with the exception that the deterministic procedure LtdBroadcast is replaced by the randomized procedure RandLtdBroadcast. The use of more randomization allows the algorithm to perform faster and drop the running time by a logarithmic factor.

RandLtdBroadcast uses a randomized procedure DecayingBroadcast(n), which guarantees that a message is transmitted to a new node with constant probability. In other words, the procedure ensures that we have some progress in broadcasting with some constant probability. Procedure DecayingBroadcast runs in parallel for every node and is displayed in detail below.

Algorithm 24: DecayingBroadcast$_v(n)$

```
for $i = 1$ to $\log n + 1$ do
    Node $v$ transmits with probability $1/2^{i-1}$
end
```

We are now in position to describe the randomized procedure RandLtdBroadcast, which again runs in parallel for all nodes. Each node v holds a label *active*(v), which is initially set to $false$.

By choosing an appropriate constant c, we can prove the following lemma about RandLtdBroadcast.

Algorithm 25: RandLtdBroadcast$_v(n, k)$

```
for $i = 1$ to $c \cdot k$ do
    if $active(v)$ then
        DecayingBroadcast$_v(n)$
    else
        if message received then
            $active(v) = \text{true}$
        end
    end
end
```

Lemma 4.3.4. *If exactly one node v is initially active,* RandLtdBroadcast(n, k) *succeeds in broadcasting the message of v to at least k nodes in time $\mathcal{O}(k \log n)$ with probability at least $1 - e^{-\alpha K}$ (α can be arbitrarily large).*

Algorithm 26: RandGossip2(ϵ)

```
execute RoundRobin $\log n$ times ;
$\delta = \epsilon / \log n$ ;
for $i = 0$ to $\log n - 1$ do
    $s = (4n/2^i) \ln(n/\delta)$ ;
    for $s_i$ times do
        Each node $v$ gets active with probability $1/n$ ;
        Each node $v$ executes RandLtdBroadcast$_v(n, 2^i)$ ;
    end
end
```

It is quite simple to show that the running time of the randomized gossiping algorithm is $\mathcal{O}(n \log^2 n \log(n/\epsilon))$. The analysis for the probability of error is similar to the analysis of RandGossip. The only difference is that we use another randomized procedure, RandLtdBroadcast, and we have to found the probability of error there as well. By using lemma 4.3.4 and the fact that the execution of RoundRobin ensures that the number of copies is large enough, we can prove that RandLtdBroadcast has error probability $o(1)$ and thus is dominated by the error probability of DistCoupColl. The algorithm can now be modified to a Las Vegas algorithm using the same idea as before and thus we have the following improvement of theorem 4.3.6.

Theorem 4.3.5. *There exists a Las Vegas algorithm that completes gossiping in any radio network of size n with expected running time $\mathcal{O}(n \log^3 n)$.*

Finally, Czumaj and Rytter [25] managed to further improve algorithm RandGossip and succeed running time of $\mathcal{O}(n \log^2 n)$. The algorithm is based on both of the two previous algorithms. The authors increased the executions of RoundRobin at the initial steps to $\mathcal{O}(\log^2 n)$ times and also replaced procedure LtdBroadcast with the randomized procedure Linear-Broadcast used for broadcasting, which is modified to run $\mathcal{O}(k)$ rounds in order to perform limited broadcasting. The analysis of the error probabilities follows the same idea as the previous algorithms, thus we omit the detailed proof. The improvements applied to RandGossip lead to the following result.

Theorem 4.3.6. *There exists a Las Vegas algorithm that completes gossiping in any radio network of size n with expected running time $\mathcal{O}(n \log^2 n)$.*

The last algorithm is the currently faster randomized gossiping algorithm. However, we do not know any non-trivial lower bound for randomized gossiping and thus the optimality of the last result is not guaranteed.

4.4 Centralized Gossiping

The main line of research for gossiping in radio networks assumes ad-hoc radio networks. Centralized gossiping was first examined by Gasieniec and Potapov [43]. The authors studied the case when only messages of *unit* size may be transmitted, i.e. a node cannot combine more than one message to a single transmission. This is clearly a more realistic assumption, especially when considering networks with a large number of nodes.

In this model, the authors first prove that gossiping in *directed* graphs requires $\Omega(n^2)$ time. They construct a simple network graph G, which is a directed line of nodes $v_0, \ldots, v_{n-1}$ extended by all edges of the form $v_j \rightarrow v_i : 0 \leq i < j \leq n-1$ (see figure 4.2). Clearly, during any successful transmission, a message can move by one position closer to the end of the line. Moreover, at any round only one node may transmit, since otherwise the reverse edges may cause a collision. Thus, the message from node v_0 needs $n-1$ rounds to reach the end node and generally node v_k needs $n-k-1$ rounds. Consequently, the total number of rounds needed will be at least $\sum_{k=0}^{n-1}(n-k-1) = \Omega(n^2)$.

They then turn their attention to gossiping in *undirected* graphs, where they propose several optimal and near-optimal algorithms working in linear time for various standard network topologies, such as rings, lines, stars and trees. As far as gossiping in general directed graphs is concerned, they showed how to perform gossiping in time $\mathcal{O}(n \log^2 n)$ and proved a lower bound of $\Omega(n \log n)$.

Gasieniec et al. [44] presented the first work to deal with centralized gossiping with arbitrarily large messages. They first show how to perform gossiping in any undirected radio network of size n in at most n rounds. The algorithm CentralizedGossiping consists of three parts: *gathering, gossiping, broadcasting.* The key idea is to first collect all messages to some connected subgraph of radius 1 (*gathering*). Then, a simple gossiping algorithm informs all the nodes of the subgraph about all the messages (*gossiping*). Finally, by reversing the order and direction of the transmissions used in the gathering stage,

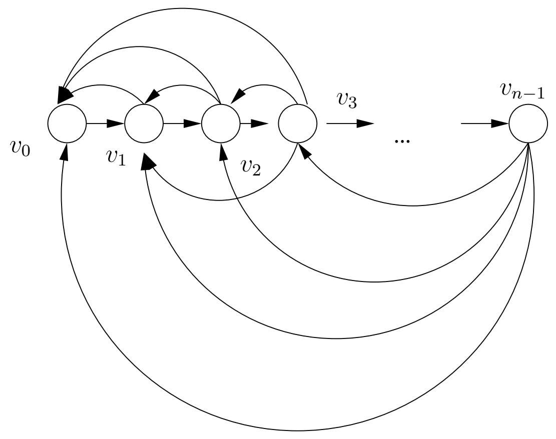


Figure 4.2: The graph constructed to prove the lower bound of $\Omega(n^2)$ for gossiping with unit size messages.

the messages collected in the subgraph are distributed to all the nodes of the network (*broadcasting*).

The main difficulty of the algorithm is how to perform the gathering stage, i.e. to collect all messages in a subgraph with radius 1. For that, we use the *2-vertex reduction principle.* Before we explain this principle in detail, let us introduce some necessary definitions. Let us assume that we have an undirected graph G with radius k, where k is the smallest integer for which there exists at least one node, such that its distance from all other nodes in the network is less or equal to k. We also choose some node c with this property, which we call a *central node.* Moreover, consider the classical partition of the graph in BFS layers with c as the root, where we denote the layer of depth i by L_i $(0 \leq i \leq k)$.

Definition 4.4.1. *A minimal covering set C_i is a subset of L_i, such that every node in L_{i+1} is connected to some node $v \in C_i$ and by removing v, this property is not preserved.*

Definition 4.4.2. *The nodes v, v', w, w' of a graph $G = (V, E)$ satisfy the* ***reduction property*** *if and only if:*

- $(v, v'), (w, w') \in E$
- $(v, w'), (w, v') \notin E$
- *The removal of v and w along with their adjacent edges does not disconnect G.*

We can now state the *2-vertex reduction principle.*

Theorem 4.4.3. *In any undirected graph G with radius $r > 1$, we can find four distinct nodes v, v', w, w' that satisfy the reduction property.*

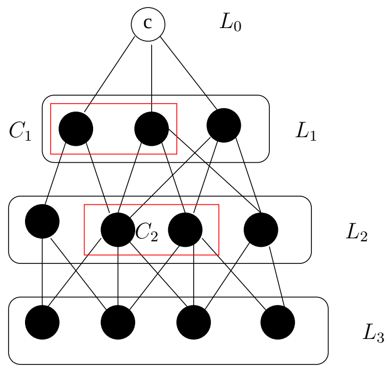


Figure 4.3: The minimal covering sets C_i $(i = 1, 2)$ for the BFS layers of a graph.

The importance of the above theorem is that we can find four distinct nodes such that the two of them are able to transmit their messages to the other two at the same round without any collisions. Furthermore, by removing the two transmitters, the graph remains connected. We are now in position to describe CentralizedGossiping in full detail.

Algorithm 27: CentralizedGossiping

```
/* Gathering stage */
while $radius > 1$ do
    In Parallel
    Select four nodes $v, v', w, w'$ that satisfy the reduction property ;
    $v$ and $w$ transmit at the same round ;
    "remove" $v$ and $w$, along with adjacent edges ;
end
/* Gossiping stage */
Consider a node $c$ of the remaining graph $G'$;
foreach $node\ v \in G'$ do
    $v$ transmits ;
end
$c$ transmits ;
/* Broadcasting stage */
Perform the transmissions of the gathering stage in the reverse order and
direction ;
```

Let us now provide some analysis for the algorithm CentralizedGossiping. The correctness of the gossiping algorithm is quite straightforward. Thus, let us concentrate on the time complexity of the algorithm. After each reduction round in the gathering stage, two nodes are removed from the graph. Thus, when the radius of the graph becomes equal 1 and assuming we need j rounds, the size of the remaining subgraph is exactly $n - 2j$.

Next, the gossiping stage needs $n - 2j - 1$ rounds so that each of the neighboring nodes sends the collected messages to c and one more round for c to disseminate all the messages to the other nodes. Finally, the broadcasting stage needs j more rounds to complete the gossiping, since we reverse the sequence and direction of the transmissions in the first stage. Summing up, algorithm CentralizedBroadcasting needs at most n rounds to complete gossiping.

Theorem 4.4.4. *Gossiping can be performed in any radio network of size n in at most n rounds.*

The authors also prove a lower bound of $\lfloor \log(n-1) \rfloor + 2$ rounds for gossiping. They show that there exists a topology where the upper and the lower bound match, i.e. gossiping can be performed in time $\lfloor \log(n-1) \rfloor + 2$. The lower bound is also nearly matched for tree topologies. Finally, the authors provide algorithms with running times dependent not on the size n, but on the parameters D and Δ of the network.

Bibliography

[1] Noga Alon, Amotz Bar-Noy, Nathan Linial, and David Peleg. A lower bound for radio broadcast. *J. Comput. Syst. Sci.*, 43(2):290–298, 1991.

[2] Reuven Bar-Yehuda, Oded Goldreich, and Alon Itai. On the time-complexity of broadcast in radio networks: An exponential gap between determinism and randomization. In *PODC*, pages 98–108, 1987.

[3] Reuven Bar-Yehuda and Amos Israeli. Multiple communication in multi-hop radio networks. In *PODC*, pages 329–338, 1989.

[4] Stefano Basagni, Imrich Chlamtac, and Danilo Bruschi. A mobility-transparent deterministic broadcast mechanism for ad hoc networks. *IEEE/ACM Trans. Netw.*, 7(6):799–807, 1999.

[5] Vartika Bhandari and Nitin H. Vaidya. On reliable broadcast in a radio network. In *PODC '05: Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*, pages 138–147, New York, NY, USA, 2005. ACM.

[6] Annalisa De Bonis, Leszek Gasieniec, and Ugo Vaccaro. Generalized framework for selectors with applications in optimal group testing. In *ICALP*, pages 81–96, 2003.

[7] Carlos Brito, Eli Gafni, and Shailesh Vaya. An information theoretic lower bound for broadcasting in radio networks. In *STACS*, pages 534–546, 2004.

[8] Danilo Bruschi and Massimiliano Del Pinto. Lower bounds for the broadcast problem in mobile radio networks. *Distributed Computing*, 10(3):129–135, 1997.

[9] Shiva Chaudhuri and Jaikumar Radhakrishnan. Deterministic restrictions in circuit complexity. In *STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 30–36, New York, NY, USA, 1996. ACM.

[10] I. Chlamtac. The wave expansion approach to broadcasting in multihop radio networks. *Communications, IEEE Transactions on*, 39(3):426–433, Mar 1991.

[11] Imrich Chlamtac and András Faragó. Making transmission schedules immune to topology changes in multi-hop packet radio networks. *IEEE/ACM Trans. Netw.*, 2(1):23–29, 1994.

[12] Imrich Chlamtac and Shay Kutten. On broadcasting in radio networks—problem analysis and protocol design. *Communications, IEEE Transactions on*, 33(12):1240–1246, Dec 1985.

[13] Bogdan S. Chlebus, Leszek Gasieniec, Alan Gibbons, Andrzej Pelc, and Wojciech Rytter. Deterministic broadcasting in unknown radio networks. In *SODA*, pages 861–870, 2000.

[14] Bogdan S. Chlebus, Leszek Gasieniec, Andrzej Lingas, and Aris T. Pagourtzis. Oblivious gossiping in ad-hoc radio networks. In *DIALM '01: Proceedings of the 5th international workshop on Discrete algorithms and methods for mobile computing and communications*, pages 44–51, New York, NY, USA, 2001. ACM.

[15] Bogdan S. Chlebus, Leszek Gasieniec, Anna Östlin, and John Michael Robson. Deterministic radio broadcasting. In *ICALP*, pages 717–728, 2000.

[16] Bogdan S. Chlebus and Dariusz R. Kowalski. Almost optimal explicit selectors. In *FCT*, pages 270–280, 2005.

[17] Malin Christersson, Leszek Gasieniec, and Andrzej Lingas. Gossiping with bounded size messages in ad hoc radio networks. In *ICALP*, pages 377–389, 2002.

[18] Marek Chrobak, Leszek Gasieniec, and Wojciech Rytter. Fast broadcasting and gossiping in radio networks. In *FOCS*, pages 575–581, 2000.

[19] Marek Chrobak, Leszek Gasieniec, and Wojciech Rytter. A randomized algorithm for gossiping in radio networks. In *COCOON*, pages 483–492, 2001.

[20] Ferdinando Cicalese, Fredrik Manne, and Qin Xin. Faster centralized communication in radio networks. In *ISAAC*, pages 339–348, 2006.

[21] Andrea E. F. Clementi, Pierluigi Crescenzi, Angelo Monti, Paolo Penna, and Riccardo Silvestri. On computing ad-hoc selective families. In *RANDOM-APPROX*, pages 211–222, 2001.

[22] Andrea E. F. Clementi, Angelo Monti, and Riccardo Silvestri. Distributed multi-broadcast in unknown radio networks. In *PODC '01: Proceedings of the twentieth annual ACM symposium on Principles of distributed computing*, pages 255–264, New York, NY, USA, 2001. ACM.

[23] Andrea E. F. Clementi, Angelo Monti, and Riccardo Silvestri. Distributed broadcast in radio networks of unknown topology. *Theor. Comput. Sci.*, 302(1-3):337–364, 2003.

[24] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition.* The MIT Press and McGraw-Hill Book Company, 2001.

[25] Artur Czumaj and Wojciech Rytter. Broadcasting algorithms in radio networks with unknown topology. In *FOCS*, pages 492–501, 2003.

[26] Anders Dessmark and Andrzej Pelc. Broadcasting in geometric radio networks. *J. Discrete Algorithms*, 5(1):187–201, 2007.

[27] Krzysztof Diks, Evangelos Kranakis, Danny Krizanc, and Andrzej Pelc. The impact of information on broadcasting time in linear radio networks. *Theor. Comput. Sci.*, 287(2):449–471, 2002.

[28] A.G. Dyachkov and V.V. Rykov. Bounds on the length of disjunctive codes. *Problemy Peredachi Informatsii*, 18(3):7–13, 1982.

[29] A.G. Dyachkov and V.V. Rykov. A survey of superimposed code theory. *Problems of Control and Information Theory*, 12:229–244, 1983.

[30] Michael Elkin and Guy Kortsarz. Logarithmic inapproximability of the radio broadcast problem. *J. Algorithms*, 52(1):8–25, 2004.

[31] Michael Elkin and Guy Kortsarz. Polylogarithmic additive inapproximability of the radio broadcast problem. *SIAM J. Discret. Math.*, 19(4):881–899, 2005.

[32] Michael Elkin and Guy Kortsarz. An improved algorithm for radio broadcast. *ACM Trans. Algorithms*, 3(1):8, 2007.

[33] Robert Elsässer, Leszek Gasieniec, and Thomas Sauerwald. On radio broadcasting in random geometric graphs. In *DISC*, pages 212–226, 2008.

[34] Yuval Emek, Leszek Gasieniec, Erez Kantor, Andrzej Pelc, David Peleg, and Chang Su. Broadcasting in udg radio networks with unknown topology. In *PODC*, pages 195–204, 2007.

[35] Yuval Emek, Erez Kantor, and David Peleg and. On the effect of the deployment setting on broadcasting in euclidean radio networks. unpublished manuscript, 2007.

[36] Zoltán Füredi. On r-cover-free families. *J. Comb. Theory Ser. A*, 73(1):172–173, 1996.

[37] Emanuele G. Fusco and Andrzej Pelc. Acknowledged broadcasting in ad hoc radio networks. *Inf. Process. Lett.*, 109(2):136–141, 2008.

[38] Iris Gaber and Yishay Mansour. Centralized broadcast in multihop radio networks. *J. Algorithms*, 46(1):1–20, 2003.

[39] Leszek Gasieniec, Dariusz R. Kowalski, Andrzej Lingas, and Martin Wahlen. Efficient broadcasting in known geometric radio networks with non-uniform ranges. In *DISC*, pages 274–288, 2008.

[40] Leszek Gasieniec and Andrzej Lingas. On adaptive deterministic gossiping in ad hoc radio networks. In *SODA*, pages 689–690, 2002.

[41] Leszek Gasieniec, Aris Pagourtzis, and Igor Potapov. Deterministic communication in radio networks with large labels. In *ESA*, pages 512–524, 2002.

[42] Leszek Gasieniec, Aris Pagourtzis, Igor Potapov, and Tomasz Radzik. Deterministic communication in radio networks with large labels. *Algorithmica*, 47(1):97–117, 2007.

[43] Leszek Gasieniec and Igor Potapov. Gossiping with unit messages in known radio networks. In *IFIP TCS*, pages 193–205, 2002.

[44] Leszek Gasieniec, Igor Potapov, and Qin Xin. Time efficient centralized gossiping in radio networks. *Theor. Comput. Sci.*, 383(1):45–58, 2007.

[45] Leszek Gasieniec, Tomasz Radzik, and Qin Xin. Faster deterministic gossiping in directed ad hoc radio networks. In *SWAT*, pages 397–407, 2004.

[46] F. K. Hwang. The time complexity of deterministic broadcast radio networks. *Discrete Appl. Math.*, 60(1-3):219–222, 1995.

[47] Piotr Indyk. Explicit constructions of selectors and related combinatorial structures, with applications. In *SODA '02: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 697–704, Philadelphia, PA, USA, 2002. Society for Industrial and Applied Mathematics.

[48] W.H. Kautz and R.R.C. Singleton. Nonrandom binary superimposed codes. *IEEE Transactions on Information Theory*, 10:363–377, 1964.

[49] János Komlós and Albert G. Greenberg. An asymptotically fast nonadaptive algorithm for conflict resolution in multiple-access channels. *IEEE Transactions on Information Theory*, 31(2):302–, 1985.

[50] Chiu-Yuen Koo. Broadcast in radio networks tolerating byzantine adversarial behavior. In *PODC*, pages 275–282, 2004.

[51] Chiu-Yuen Koo, Vartika Bhandari, Jonathan Katz, and Nitin H. Vaidya. Reliable broadcast in radio networks: the bounded collision case. In *PODC*, pages 258–264, 2006.

[52] Dariusz Kowalski and Andrzej Pelc. Optimal deterministic broadcasting in known topology radio networks. *Distributed Computing*, 19(3):185–195, January 2007.

[53] Dariusz R. Kowalski and Andrzej Pelc. Broadcasting in undirected ad hoc radio networks. In *PODC '03: Proceedings of the twenty-second annual symposium on Principles of distributed computing*, pages 73–82, New York, NY, USA, 2003. ACM.

[54] Dariusz R. Kowalski and Andrzej Pelc. Faster deterministic broadcasting in ad hoc radio networks. In *STACS '03: Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science*, pages 109–120, London, UK, 2003. Springer-Verlag.

[55] Dariusz R. Kowalski and Andrzej Pelc. Time of deterministic broadcasting in radio networks with local knowledge. *SIAM J. Comput.*, 33(4):870–891, 2004.

[56] Dariusz R. Kowalski and Andrzej Pelc. Time complexity of radio broadcasting: adaptiveness vs. obliviousness and randomization vs. determinism. *Theor. Comput. Sci.*, 333(3):355–371, 2005.

[57] Evangelos Kranakis, Danny Krizanc, and Andrzej Pelc. Fault-tolerant broadcasting in radio networks. *J. Algorithms*, 39(1):47–67, 2001.

[58] Eyal Kushilevitz and Yishay Mansour. An $\omega(d \log(n/d))$ lower bound for broadcast in radio networks. *SIAM J. Comput.*, 27(3):702–712, 1998.

[59] Ding Liu and Manoj Prabhakaran. On randomized broadcasting and gossiping in radio networks. In *COCOON*, pages 340–349, 2002.

[60] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[61] Gianluca De Marco. Distributed broadcast in unknown radio networks. In *SODA*, pages 208–217, 2008.

[62] Gianluca De Marco and Andrzej Pelc. Faster broadcasting in unknown radio networks. *Inf. Process. Lett.*, 79(2):53–56, 2001.

[63] Rajeev Motwani and Prabhakar Raghavan. Randomized algorithms. In *The Computer Science and Engineering Handbook*, pages 141–161. 1997.

[64] Elena Pagani and Gian Paolo Rossi. Reliable broadcast in mobile multihop packet networks. In *MobiCom '97: Proceedings of the 3rd annual ACM/IEEE international conference on Mobile computing and networking*, pages 34–42, New York, NY, USA, 1997. ACM.

[65] Andrzej Pelc. Fault-tolerant broadcasting and gossiping in communication networks. *Networks*, 28(3):143–156, 1996.

[66] Andrzej Pelc and David Peleg. Feasibility and complexity of broadcasting with random transmission failures. In *PODC '05: Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*, pages 334–341, New York, NY, USA, 2005. ACM.

[67] David Peleg. Time-efficient broadcasting in radio networks. In *DISC*, pages 3–4, 2007.

[68] Krishnamurthi Ravishankar and Suresh Singh. Broadcasting on [0,l]. In *Proceedings of the international workshop on Broadcasting and gossiping 1990*, pages 299–319, New York, NY, USA, 1994. Elsevier North-Holland, Inc.

[69] Krishnamurthi Ravishankar and Suresh Singh. Asymptotically optimal gossiping in radio networks. *Discrete Appl. Math.*, 61(1):61–82, 1995.

[70] Krishnamurthi Ravishankar and Suresh Singh. Gossiping on a ring with radios. volume 6, pages 115–126. World Scientific Publishing, Singapore, 1996.

[71] Arunabha Sen and Mark L. Huson. A new model for scheduling packet radio networks. *Wireless Networks*, 3(1):71–82, 1997.

[72] Jiro Uchida, Wei Chen, and Koichi Wada. Acknowledged broadcasting and gossiping in ad hoc radio networks. *Theor. Comput. Sci.*, 377(1-3):43–54, 2007.

[73] Ying Xu. An $\mathcal{O}(n^{1.5})$ deterministic gossiping algorithm for radio networks. *Algorithmica*, 36(1):93–96, 2003.