The impact of knowledge on broadcasting time in radio networks

Krzysztof Diks * Evangelos Kranakis † Danny Krizanc † Andrzej Pelc ‡

Abstract

We consider the problem of distributed deterministic broadcasting in radio networks. Nodes send messages in synchronous time-slots. Each node v has a given transmission range. All nodes located within this range can receive messages from v. However, a node situated in the range of two or more nodes that send messages simultaneously, cannot receive these messages and hears only noise. Each node knows only its own position and range, as well as the maximum of all ranges. Broadcasting is adaptive: Nodes can decide on the action to take on the basis of previously received messages, silence or noise. We prove lower bounds on broadcasting time in this model and construct broadcasting protocols whose performance nearly matches these bounds for the simplest case when nodes are situated on a line. We also show that if nodes do not even know their own range, every broadcasting protocol must be hopelessly slow.

While distributed randomized broadcasting algorithms, and, on the other hand, deterministic off-line broadcasting algorithms assuming full knowledge of the radio network, have been extensively studied in the literature, ours are the first results concerning broadcasting algorithms that are distributed and deterministic at the same time. We show that in this case the amount of knowledge available to nodes influences the efficiency of broadcasting in a significant way.

keywords: broadcasting, distributed, deterministic, radio network.

*Instytut Informatyki, Uniwersytet Warszawski, Banacha 2, 02-097 Warszawa, Poland. E-mail: diks@mimuw.edu.pl
Partially sponsored by KBN grant 8T11 C 03614. This work was done during the author's stay at the Université du Québec à Hull.
†School of Computer Science, Carleton University, Ottawa, Ontario, K1S 5B6, Canada.
E-mail: {kranakis,krizanc}@scs.carleton.ca
Research supported in part by NSERC (Natural Sciences and Engineering Research Council of Canada) grant.
‡Département d'Informatique, Université du Québec à Hull, Hull, Québec J8X 3X7, Canada. E-mail: Andrzej_Pelc@uqah.uquebec.ca
Research supported in part by NSERC grant OGP 0008136.

1 Introduction

Radio communication networks have recently received growing attention. This is due to the expanding applications of radio communication, such as cellular phones and wireless local area networks. The relatively low cost of infrastrucure and the flexibility of radio networks make them an attractive alternative to other types of communication media.

A radio network is a collection of transmitter-receiver devices (referred to as *nodes*). Nodes send messages in synchronous time-slots. Each node v has a given transmission range. All nodes located within this range can receive messages from v. However, a node situated in the range of two or more nodes that send messages simultaneously, cannot receive these messages and hears only noise.

One of the fundamental tasks in network communication is *broadcasting*. One node of the network, called the *source*, has a piece of information which has to be transmitted to all other nodes. Remote nodes get the source message via intermediate nodes, in several hops. One of the most important performance parameters of a broadcasting scheme is the total time it uses to inform all nodes of the network.

1.1 Previous work

In most of the research on broadcasting in radio networks [1, 4, 6] the network is modeled as an undirected graph in which nodes are adjacent if they are in the range of each other. A lot of effort has been devoted to finding good upper and lower bounds on the broadcast time in radio networks represented as arbitrary graphs, under the assumption that nodes have full knowledge of the network. In [1] the authors proved the existence of a family of n-node networks of radius 2, for which any broadcast requires time $\Omega(\log^2 n)$, while in [4] it was proved that broadcasting can be done in time $O(D + \log^5 n)$ for any n-node network of diameter D. In [11] the authors restricted attention to communication graphs that can arise from actual geometric locations of nodes in the plane. They proved that scheduling optimal broadcasting is NP-hard even when restricted to such graphs and gave an $O(n \log n)$ algorithm to schedule an optimal broadcast when nodes are situated on a line. In [5] the authors discussed fault-tolerant broadcasting in radio networks arising from geometric locations of nodes on the line and in the plane. On the other hand, in [2] a randomized protocol was given for arbitrary radio networks where nodes have no topological knowledge of the network, not even about neighbors. This randomized protocol runs in expected time $O(D \log n + \log^2 n)$.

1.2 Our results

The novelty of our approach consists in considering broadcasting protocols that are distributed and deterministic at the same time. We assume that nodes have only local knowledge concerning their own position and range and additionally they know the maximum R of all ranges. This is a realistic assumption, as the transmitter-receiver devices can have varying power but usually belong to a set of *a priori* known standard types. Our aim is to show to what extent this restriction of knowledge concerning the network affects efficiency of broadcasting. We consider the simplest scenario when nodes are situated at integer points on the line. We prove the lower bound $\Omega(\frac{\log^2 R}{\log\log R})$ on broadcasting time of any deterministic protocol. (This lower bound is of course also valid for nodes situated in the plane.) Moreover, we show two broadcasting protocols: one running in time $O(D\frac{\log^2 R}{\log\log R})$ and the other in time $O(D+\log^2 R)$, where D is the *depth* of the communication graph, i.e., the maximum length of a shortest path from the source to any node. Thus our protocols are asymptotically optimal for constant D and for $D = \Omega(\log^2 R)$. We also consider the extreme scenario when nodes do not even know their own range. Under this assumption we show that every broadcasting protocol must use time $\Omega(R)$ for some networks of depth 2.

The paper is organized as follows. In section 2 we fix our terminology and describe the model. Section 3 is devoted to the lower bound in case when nodes do not even know their own range. In section 4 we show that if each node knows its own range, this lower bound can be dramatically invalidated. In section 5 we prove a matching lower bound under the assumption from section 4. Section 6 contains conclusions and open problems.

2 Preliminaries and model description

Nodes are situated at integer points of the line and are identified with the respective integers. Every node v has a non-negative integer *range* $r(v)$. The set of pairs $(v, r(v))$, for all nodes v, with a distinguished node s called the *source*, is referred to as a *configuration*. If v sends a message, the transmission reaches exactly those nodes that are in the segment $[v - r(v), v + r(v)]$. These nodes are said to be in the range of v. However, a node situated in the range of two or more nodes that send messages simultaneously, cannot receive these messages and hears only noise.

Actions of nodes are performed in synchronous time-slots. We consider two models. In the main model (considered in sections 4 and 5) the *a priori* knowledge of every node v consists of its position v, its range $r(v)$ and the maximum R over all ranges. It is important to stress that nodes do not know positions or ranges of any other nodes. In the second model (considered in section 3) this knowledge is further reduced: a node v does not even know $r(v)$. We use the latter scenario to prove a strong lower bound on broadcasting time.

In each time-slot a node v receives one of the following inputs: either silence, (when neither itself nor any other node in whose range v is situated transmits), or a message (if a unique node in whose range v is situated transmits), or noise (if v is situated in the ranges of at least two simultaneously transmitting nodes). All nodes run a common broadcasting protocol. Broadcasting is *adaptive*: Every node can compute its action to be performed in a given time slot on the basis of previously received inputs. This action is either sending a particular message or keeping silent.

The *reachability graph* associated with a given configuration is the directed graph G whose vertices are nodes of the configuration and there is a directed edge from v to w, if w is in the range of v. We assume that there exists a directed path from the source s to any node of G. Let $d(v)$

denote the length of the shortest directed path in G, from s to v. The *depth* D of the graph G (or of the underlying configuration) is defined as the maximum of $d(v)$ over all nodes v of the configuration. The set of all nodes of a configuration can be partitioned into $D+1$ *layers* $L_0, ..., L_D$, where $L_i = \{v : d(v) = i\}$. Clearly, D is a lower bound on broadcasting time of any protocol. On the other hand, if all nodes know the positions and ranges of all other nodes, i.e., the entire configuration, it is easy to construct a distributed deterministic broadcasting protocol working in time $O(D)$.

3 Lower bound under range ignorance

We begin by considering the extreme scenario when the *a priori* knowledge of each node is limited to its position and the maximum R over all ranges. (A node does not even know its own range.) Under this assumption we show that the worst case broadcasting time is $\Omega(R)$ for some configurations of constant depth.

We consider the following situation: $n+1$ nodes are located at the points $0, 1, \ldots, n$ on the line. We assume that n is divisible by 3. Node 0 is the source. It has range equal to $2n/3$. $k \geq 1$ out of the nodes $\{1, 2, \ldots, n/3\}$ have range equal to n. For the remaining nodes in $\{1, 2, \ldots, n/3\}$, node i has range equal to $2n/3 - i$. The nodes $\{n/3+1, \ldots, n\}$ all have range equal to 1. Thus $R = n$ and $D = 2$.

We will say a node in $\{1, \ldots, n/3\}$ is *strong* if its range is n, *weak* otherwise. There are $2^{n/3} - 1$ possible configurations, corresponding to the possible settings of weak and strong nodes.

The protocol which has the nodes $0, 1, \ldots, n/3$ broadcast in succession gives an upper bound of $n/3+1$ on broadcasting time. The remainder of this section is devoted to showing that any protocol must use at least $n/3+1$ steps, i.e., the above protocol is optimal.

In order to show this lower bound we must be more precise in our model of a protocol. We assume that the $n+1$ nodes are universal Turing machines running synchronously using a global clock initially set to 0. The input to a node i in $1, \ldots, n$ is a program P_i. Node 0 has as input P_0 and a string M on a special input tape that is to be broadcast. At the completion of the protocol, all nodes will have entered a terminal state and will have output M onto a special output tape. All steps of a protocol (except the first) consist of three phases: Receive, Compute, Broadcast. During the Receive phase, every node v reads from a special reception input tape the results of the Broadcast phase of the previous step which is determined by the rules for packet radio networks and in whose range v is situated in the given configuration (see the discussion below concerning the Broadcast phase). During the Compute phase, the nodes perform an arbitrary computation based upon the input they received, their current state (including the contents of all tapes) and the program they are running. As a result of this computation each node decides on one of two actions, either to broadcast or be silent during the Broadcast phase. If a processor decides to broadcast then it writes its state including the contents of all of its tapes, the positions of its heads, etc., to a special broadcast output tape. If it decides not to broadcast it writes a special symbol indicating "silence" to the broadcast output tape. After a Broadcast phase, a node's v reception input tape contains one of the following:

1. a special symbol representing "silence", if none of the nodes in whose range v is situated decided to broadcast;
2. a special symbol representing "noise", if two or more of the nodes in whose range v is situated decided to broadcast;
3. the contents of the broadcast tape of a node w, if w is the unique node that decided to broadcast among nodes in whose range v is situated.

Recall that every node is in its own range. The first step has no Receive phase. If a node enters a terminal state it no longer participates in the protocol. It is assumed that the same programs $P_0, \ldots, P_n$ are used for all input configurations and for all messages M.

We are now ready to state the main theorem of this section:

Theorem 3.1 *For any deterministic broadcast protocol there exists a configuration on which it requires $n/3+1$ steps.*

Proof: Assume to the contrary there exists a protocol $\mathcal{P}$ which on all configurations finishes in $t \leq n/3$ steps. Let $|P_i|$ be the number of bits required to describe the input program to node i for $\mathcal{P}$. Let M be a string with Kolmogorov complexity [8] greater than $n+\sum_{i=0}^{n}|P_i|$ bits. The intuitive reason for choosing a message of this complexity is that it precludes the possibility of encoding it in at most $n/3$ time-slots by silence-noise bits.

The proof of the theorem is based on the following three lemmas.

Lemma 3.1 *For input broadcast message M, for all configurations, there exists at least one step of protocol $\mathcal{P}$ during which precisely one strong node broadcasts.*

Proof: Note that if in each step of the protocol either zero or more than one strong nodes broadcast, only t bits are required to encode the effect of the first $n/3+1$ processors on the last $n/3$ processors. The proof of the lemma is by contradiction and is immediate from following claim:

Claim. *Given the programs $P_{2n/3+1}, \ldots, P_n$ and the $t \leq n/3$ bits indicating whether zero strong nodes or more than one strong nodes broadcast in step j, for $j = 1, \ldots, t$, it is possible for a Turing machine to simulate the actions of nodes $2n/3+i, \ldots, n$ for i steps.*

The proof of the claim is by induction. Clearly, the programs are sufficient to simulate the first step of each node in $\{2n/3+1, \ldots, n\}$. Assume that the first i steps have been simulated correctly for nodes $2n/3+i, \ldots, n$. If the ith bit indicates that two or more strong nodes broadcast on step i then the reception input tapes of all nodes contain the special "noise" symbol. If on the other hand the ith bit indicates no strong nodes broadcast, the reception input tape of node j depends only on the actions and states of nodes $j-1$ and $j+1$ (if it exists) which are known for $j = 2n/3+i, \ldots, n$. I.e., it is possible to simulate the actions of all nodes $2n/3+i+1, \ldots, n$ during step $i+1$.

Clearly, using the claim and the above description of the broadcast problem it is possible to construct a program for M using fewer than $t + \log n + \sum_{i=2n/3+1}^{n} |P_i|$ bits by simulating the actions of node n for $t \leq n/3$ steps and then outputting the contents of its special output tape. □

The second lemma shows that the source 0 must take one step to broadcast alone.

Lemma 3.2 *For input broadcast message M, for all configurations, there exists at least one step of protocol $\mathcal{P}$ during which which node 0 broadcasts and precisely zero nodes among $1, \ldots, n/3$ broadcast.*

Proof: The proof is similar to that of Lemma 3.1. In this case it is possible to simulate nodes $1, \ldots, n/3 - i + 1$ for i steps given their programs and a bit string indicating whether or not node 0 broadcasts on step i on input M. □

The third lemma shows that the actions taken by nodes $1, \ldots, n/3$ for the first $t \leq n/3$ steps of any protocol are the same for all configurations.

Lemma 3.3 *For input broadcast message M, at the end of step $i \leq n/3$ of protocol $\mathcal{P}$, the state of each of the nodes $0, 1, \ldots, 2n/3 - i + 1$ (including the contents of their broadcast tape) is the same for all configurations.*

Proof: The proof is by induction. Since all of the programs are the same for all configurations, the Compute and Broadcast phases of step 1 performed by all nodes on input M are the same for all configurations, i.e., the lemma holds for step 1. Assume the lemma holds for i steps and consider what happens in step $i+1 \leq n/3$. For all nodes, their reception input tape depends only upon contents of the broadcast tapes of nodes in whose range they are situated. For nodes $0, 1, \ldots, 2n/3 - i$ the contents of these tapes is the same for all configurations, since all nodes in whose range they are situated are among the nodes $0, 1, \ldots, 2n/3 - i + 1$. Given that they run the same program for all configurations it follows that the state of nodes $0, 1, \ldots, 2n/3 - i$ (including the contents of their broadcast tape) is the same for all configurations at the end of step $i+1$. □

As a consequence of Lemma 3.3, for protocol $\mathcal{P}$ running in $t \leq n/3$ steps on broadcast input message M, for each of the nodes $0, 1, \ldots, n/3$, its actions, i.e., whether it broadcasts or is silent, is the same during each of the steps of the protocol, for all configurations. Thus, we can consider the actions of these nodes as consisting of t sets, $S_1, S_2, \ldots, S_t$ where set S_i is the subset of these nodes that broadcast during step i. We are now ready to complete the proof of the theorem.

Consider the protocol $\mathcal{P}$ on broadcast input message M. By lemma 3.2 one of the t steps of $\mathcal{P}$ must have node 0 broadcast while all nodes in $1, \ldots, n/3$ are silent. Consider the remaining $t-1 < n/3$ steps, the only ones during which the nodes $1, \ldots, n/3$ can broadcast. Assume that none of the sets associated with these steps are singletons. Then for the configuration consisting of all strong nodes, in all steps, either all strong nodes are silent or two or more strong nodes broadcast. This contradicts lemma 3.1. Therefore there must be at least one singleton set. For all singleton sets, assign the weak range to the node in the set. Now remove all nodes assigned weak range from the sets. If after this process, no singletons are created, then the configuration with all remaining nodes assigned the strong range again contradicts Lemma 3.1. Assign the resulting singletons weak range, and continue with this process. It must stop with all sets having been reduced to singletons or to the empty set. Since $t-1 < n/3$, there exists at least one node which does not appear in any singleton. Consider the configuration where that node is strong and all others are weak. In this configuration, the given node never broadcasts and therefore Lemma 3.1 is again contradicted. Therefore, no such protocol $\mathcal{P}$ exists and at least $t+1$ steps are required to solve the problem. $\square$

It easily follows from considerations in the next section that in case when each node knows its position and range, as well as the maximum R over all ranges, broadcasting for configurations considered above can be done in time $O(\log R)$. (See Algorithm 1 – leader election in a cluster.)

4 Broadcast protocols with known range

In this section we show that if every node knows its own range and position, as well as the maximum R over all ranges, the lower bound from section 3 can be dramatically invalidated. We first show a broadcast protocol running in worst-case time $O(\frac{\log^2 R}{\log\log R})$, for all configurations of depth 2. (Recall that without knowledge of nodes' own range we showed such a configuration requiring time $\Omega(R)$.) This protocol can be generalized to give time $O(D\frac{\log^2 R}{\log\log R})$, for all configurations of depth D. We also show another protocol which runs in worst-case time $O(D + \log^2 R)$ and thus improves over the performance of the previous one for configurations of large depth. The lower bound to be proved in section 5 shows that the above protocols are asymptotically optimal for constant D and for $D = \Omega(\log^2 R)$, respectively.

For any nonempty set S of nodes, a set $S' \subseteq S$ is *right-equivalent* to S, if $\max\{v + r(v); v \in S'\} = \max\{v + r(v); v \in S\}$. *Left-equivalent* subsets are defined similarly. We will first restrict our considerations to informing nodes larger than the source, and we will use the term *equivalent* instead of right-equivalent.

For simplicity we assume that R is a power of 2. Modifications in the general case are obvious. For every integer j and every $l = 0, 1, ..., \log R$, we define $I(j,l) = \{j2^l, j2^l + 1, ..., (j+1)2^l - 1\}$. Fix a layer L. Assume, without loss of generality, that all ranges of nodes in L are strictly positive. The *cluster* $C(j,l)$ is defined as the set of nodes $\{v \in L \cap I(j,l) : 2^l \leq r(v) < 2^{l+1}\}$, if this set is nonempty. The integer l is called the level of the cluster $C(j,l)$. A node $v \in L$ belongs to the cluster $C(j(v), l(v))$, where $l(v) = \lfloor \log r(v) \rfloor$ and $j(v) = \max\{j : j2^{l(v)} \leq v\}$.

Lemma 4.1 *Clusters form a partition of L. Every pair of nodes in a cluster are in each others range.*

The *leader* of a cluster is its node v with maximum value $v + r(v)$. If there are many such nodes then the leader is the one with maximum range among them. Notice that the singleton set of a leader is equivalent to the cluster.

A node $u \in X \subseteq L$ is called X-*nonessential* if there exists $v \in X$ such that u is in the range of v and either (1) $v + r(v) > u + r(u)$ or (2) $v + r(v) = u + r(u)$ and $v < u$.
The set of all nodes X-nonessential is denoted by X^- and the set of all other nodes in X (called X-essential) is denoted by X^+.

Lemma 4.2 *The set X^+ is equivalent to X.*

4.1 A $O(D\frac{\log^2 R}{\log\log R})$ broadcasting protocol

We first construct a broadcasting protocol working in time $O(\frac{\log^2 R}{\log\log R})$, for configurations of depth 2. To this end we will solve the following problem. Consider a layer $L \subseteq \{0, ..., R-1\}$, for which we have $v + r(v) \geq R$, for any node $v \in L$. We want to construct a small subset of L equivalent to L. We will show how to construct such a set of size $O(\log R)$ in time $O(\frac{\log^2 R}{\log\log R})$. This will yield the desired protocol for configurations of depth 2, with all nodes in this small equivalent subset broadcasting sequentially.

Lemma 4.3 *For every level $l = 0, 1, ..., \log R$, the set L contains at most two (consecutive) clusters $C(2^{\log R - l} - 2, l)$ and $C(2^{\log R - l} - 1, l)$.*

It follows that the number of clusters in L is at most $2\log R + 2$. The set of leaders of all clusters is equivalent to L. In fact, the small equivalent set that we seek, is the subset of the set of all leaders.

Let L_e (L_o) be the set of those nodes $u \in L$ for which $j(u)$ is even (odd).

Lemma 4.4 *The set $L_e^+ \cup L_o^+$ is equivalent to L and has size $O(\log R)$.*

We now show how to construct L_o^+. The construction of L_e^+ is similar. Consider clusters $C(j, l)$ with odd j.

Lemma 4.5 *Let $l_1 \geq l_2 + 3$. If there exists a node $v \in C(j_1, l_1)$ such that $v \geq R - 3 \cdot 2^{l_1 - 3} + 1$ then all nodes in $C(j_2, l_2)$ are L_o-nonessential.*

Proof: $v + r(v) \geq R - 3 \cdot 2^{l_1-3} + 1 + 2^{l_1} = R + 5 \cdot 2^{l_1 - 3} + 1$, and, for all $u \in C(j_2, l_2)$, we have $u + r(u) \leq R - 1 + 2^{l_2+1} - 1 \leq R + 2^{l_1-2} - 2 = R + 2 \cdot 2^{l_1-3} - 2$. □

The cluster $C(j_2, l_2)$ satisfying Lemma 4.5 is called *useless*. All other clusters are called *useful.*

Lemma 4.6 *Let $l_1 \geq l_2 + 3$. If $v \in C(j_1, l_1)$ and $v < R - 3 \cdot 2^{l_1 - 3} + 1$ then v is not in the range of any node from $C(j_2, l_2)$.*

Proof: The left-most node in $C(j_2, l_2)$ is $R - 2^{l_2}$. The left-most node in the range of it is $R - 2^{l_2} - 2^{l_2+1} + 1 = R - 3 \cdot 2^{l_2} + 1 \geq R - 3 \cdot 2^{l_1-3} + 1$. □

A sequence of clusters $\gamma = (C(j_1, l_1), ..., C(j_s, l_s))$ is called a *chain* if it satisfies the following conditions:

1. $l_1 > ... > l_s$,

2. for all i, all nodes from clusters of levels $l_i, l_{i+1}, ..., l_s$ are in the range of all nodes in cluster $C(j_i, l_i)$ but no node from clusters of levels $l_1, ..., l_{i-1}$ is in the range of a node from cluster $C(j_i, l_i)$.

Lemma 4.7 *Useful clusters in layer L_o can be partitioned into three chains.*

Proof: It follows from Lemmas 4.5 and 4.6 that γ_0, γ_1, γ_2 is such a partition, where γ_i is the sequence of consecutive useful clusters on levels $l \bmod 3 = i$. □

We can now formulate a high-level description of an algorithm constructing L_o^+. It consists of three phases.

Phase 1. Find and eliminate useless clusters.

Phase 2. In every chain γ_i, for $i = 0, 1, 2$, find leaders of clusters from this chain.

Phase 3. Eliminate nonessential nodes from the set of leaders found in phase 2.

Phase 1 proceeds in $\log R$ steps numbered $0, 1, ..., \log R - 1$. In step i, nodes from cluster $C(j, \log R - i)$ larger than $R - 3 \cdot 2^{\log R - i - 3}$ transmit a (arbitrary) message. Every node of level at most $\log R - i - 3$, that got such a signal, knows that it belongs to a useless cluster and stops. Correctness of phase 1 follows immediately from Lemmas 4.5 and 4.6.

Phase 2 is the most difficult. We show how to find leaders of all clusters in a chain in time $O(\frac{\log^2 R}{\log \log R})$. Define $b(j, l) = (j + 1)2^{l+1} - 1$. We assign to every node $v \in C(j, l)$ its *label* defined as follows:

$$lab(v) = (v + r(v) - b(j, l) - 1)R + (b(j, l) - v).$$

Notice that $0 \leq lab(v) \leq R^2 - R$.

Lemma 4.8 *Different nodes in a cluster have different labels. A node v is a leader in $C(j, l)$, if and only if, $lab(v) > lab(u)$, for all nodes u in $C(j, l)$, different from v.*

Every node v can compute parameters $j(v)$ and $l(v)$, as well as its label $lab(v)$ knowing its position and range. Labels will be used to elect a leader by binary search. Upon completion of the algorithm the value of a boolean variable leader(v) informs the node v if it is a leader.

Algorithm 1: Election of a leader in a cluster – explicit binary search. Algorithm for node v.

$l := 0$; $r := R^2 - R$;
while $r - l > 0$ **do**
  $m := \lfloor (l+r)/2 \rfloor$;
  if $m + 1 \leq lab(v) \leq r$ **then** broadcast /*/
  else keep silent;
  if silence **then** $r := m$
  else $l := m + 1$;
leader(v):=$(lab(v) = l)$;
if leader(v) **then** broadcast $(v, r(v))$. /**/

In step /*/ the node broadcasts any message (it is sufficient just to send a signal). In step /**/ messages may be different, depending on the purpose the leader is used for. In our case we want to identify nonessential leaders in clusters of the chain. Hence the leader broadcasts $(v, r(v))$.

For all nodes $v \in C(j, l)$, values of l and r are the same after each turn of the **while** loop. If v is a leader, we have $l \leq lab(v) \leq r$. Hence a leader in a cluster can be elected in time $\Theta(\log R)$. Algorithm 1 could be used to elect leaders in each cluster of the chain separately. However, this would result in time $\Theta(\log^2 R)$. In order to speed up the process, we need to elect leaders in many clusters simultaneously. However, in doing so, we need to avoid interference among nodes from different clusters broadcasting at the same time.

We start with a generalization of Algorithm 1. $P = R^2 - R + 1$ is the number of possible labels of nodes. Let $S \geq \lceil \log P \rceil$ be an integer and let A be an arbitrary set of size P of binary sequences of length S. Denote by $\alpha_0, \alpha_1, ..., \alpha_{P-1}$ the lexicographic ordering of sequences from A. Assign to every node u its *binary label*: $binlab(u) = \alpha_{lab(u)}$. Clearly $binlab(u) > binlab(v)$, if and only if, $lab(u) > lab(v)$. For a sequence α, $pref(\alpha, i)$ denotes its prefix of length i, and $\alpha[i]$ denotes the ith term of α. In particular, $pref(\alpha, 0)$ is the empty sequence ϵ.

Suppose that every node knows its binary label with respect to a given set A. The following algorithm elects a leader in a cluster, using binary labels.

Algorithm 2: Election of a leader in a cluster – implicit binary search. Algorithm for node v.

$\beta := \epsilon$; $\alpha := binlab(v)$;
for $i := 1$ **to** S **do**
  if $pref(\alpha, i-1) = \beta$ and $\alpha[i] = 1$ **then** broadcast
  else keep silent;
  if silence **then** $\beta := \beta \bullet 0$
  else $\beta := \beta \bullet 1$;
leader(v):=$(\beta = \alpha)$;

if leader(v) **then** broadcast $(v, r(v))$.

It can be easily shown by induction on the number of iterations of the **for** loop that the sequence β is the same for all $v \in C(j,l)$, and that $\beta = pref(binlab(v), i)$, if v is a leader. The correctness of Algorithm 2 follows from this observation. Moreover, if $S = \lceil \log P \rceil$ and A is the set of all binary sequences of length S, Algorithm 2 is a restatement of Algorithm 1.

We will now use Algorithm 2 to find leaders in all clusters of a chain γ simultaneously. In each cluster a separate "copy" of the algorithm will perform election. Notice that if no node broadcasts in a given step, we can extend the sequence β by one bit: 0, in every cluster. Clearly, in nontrivial cases, exclusively silent steps cannot accomplish leader election. Nevertheless, we will keep the number of "noisy" steps small, and at the same time elect leaders fast. Noisy steps are a problem because nodes from a given cluster can be heard in all clusters of lower levels. In order to prevent this interference from disturbing computations in clusters of lower levels, we add, for each step of Algorithm 2 performed in any cluster, two steps verifying if noise heard by nodes in this cluster is not caused by nodes from clusters of higher levels. If it is, nodes from this cluster repeat the same step of Algorithm 2 and we say that the cluster is *delayed*. If we guarantee that each cluster is delayed only during $O(\frac{\log^2 R}{\log\log R})$ steps and that Algorithm 2 works in time $O(\frac{\log^2 R}{\log\log R})$ (i.e., $S = O(\frac{\log^2 R}{\log\log R})$) then leaders in all clusters will be elected in time $O(\frac{\log^2 R}{\log\log R})$.

To this end we show that the set A of sequences can be chosen to make the number of "noisy" steps $O(\frac{\log R}{\log\log R})$. Then no cluster will be delayed more than $O(\frac{\log^2 R}{\log\log R})$ times.

Let $S = \lceil \frac{2\log^2 P}{\log\log P} \rceil$ and $H = \lceil \frac{\log P}{\log\log P} \rceil$.

Lemma 4.9 *There exist at least P binary sequences of length S containing at most H terms 1.*

Proof: The number of such sequences is at least

$$\binom{S}{H} \geq (S/H)^H \geq P.$$

□

Thus we can take as A any set of P binary sequences of length S containing at most H terms 1. It remains to show, how nodes that hear noise can determine that it is caused by broadcasting nodes from clusters of higher levels. Suppose that nodes know the consecutive number of their cluster in the chain. Fix a time unit i in which various steps of Algorithm 2 are performed in various clusters. In time unit $i+1$ (the first verifying step), all nodes that heard noise in time unit i and are in clusters with even number, broadcast. In time unit $i+2$ (the second verifying step), all nodes that heard noise in time unit i and are in clusters with odd number, broadcast. Notice that if some cluster heard noise in time unit i, caused by a higher level cluster, it must hear noise in *both* verifying steps. Such clusters repeat the step of Algorithm 2 performed in time unit i. On the other hand, clusters that hear noise in only one verifying step, can perform the next step of

Algorithm 2 because all nodes of this cluster know that noise heard in time unit i was caused by nodes from this cluster. In order to keep synchrony, nodes that hear nothing in time unit i, perform the corresponding step of Algorithm 2 and wait two time units.

In the above argument we assumed that nodes know the number of their cluster in the chain. This knowledge is easy to get. Clusters in a chain can be numbered in time $O(\log R)$ as follows. In the jth step, nodes from cluster of level $\log R - j + 1$ broadcast (any signal). (Recall that only useful clusters remain at this point.) Every node v counts how many clusters of higher levels have broadcast. This number plus 1 is the consecutive number of the cluster of node v in the chain. Clearly, each node knows to which chain γ_i its cluster belongs, because it knows the level of its cluster. This yields the following lemma.

Lemma 4.10 *It is possible to elect leaders in all clusters of a chain in time* $O(\frac{\log^2 R}{\log\log R})$.

Thus phase 2 can be performed in time $O(\frac{\log^2 R}{\log\log R})$. Phase 3 can be done simultaneously with phase 2. After its election, every leader v broadcasts the message $(v, r(v))$. Using this information every leader can deduce if it is essential or not.

Thus we have shown how to construct a set of size $O(\log R)$ equivalent to the layer L, in time $O(\frac{\log^2 R}{\log\log R})$. This implies that broadcasting in configurations of depth 2 can be done in time $O(\frac{\log^2 R}{\log\log R})$. This result can be easily generalized as follows.

Theorem 4.1 *Broadcasting in a configuration of depth D can be done in time* $O(D\frac{\log^2 R}{\log\log R})$.

Proof: Let $L_0, L_1, ..., L_D$ be layers of a configuration. In the first step the source s broadcasts the source message together with its position and range. Nodes of the first layer are either in the right segment $\{s+1, ..., s+r(s)\}$ or in the left segment $\{s-r(s), ..., s-1\}$, and every node knows in which of them it is situated. For the right segment we find a set of nodes of size $O(\log R)$ equivalent to the set of those nodes v from this segment for which $v + r(v) > s + r(s)$. We proceed similarly and separately for the left segment. In order to find these sets we use the above described algorithm, shifting each segment to make its right end equal to $R - 1$. The left segment is handled similarly. During the election of leaders, every leader already knows the source message. When it is elected it broadcasts this message together with the number of its layer, its position and range. Using this information, every node of layer L_1 can compute to which segment (to the right or to the left of the source) it belongs and continue broadcasting as above. □

4.2 A $O(D + \log^2 R)$ broadcasting protocol

We now construct a broadcasting protocol working in worst-case time $O(D + \log^2 R)$ and hence outperforming the previous one for D of order higher than $\log\log R$. We first carry out the reasoning under assumptions that all nodes are larger than the source and that every node knows its layer number. At the end we show how these assumptions can be eliminated.

Lemma 4.11 *Let $C(j_1, l)$ and $C(j_2, l)$ be two clusters of the same level l, such that $j_2 - j_1 > 4$. Then there does not exist a node situated both in the range of some node in $C(j_1, l)$ and of some node in $C(j_2, l)$.*

Proof: The rightmost node in the range of a node from $C(j_1, l)$ is $(j_1 + 3) \cdot 2^l - 2 < (j_2 - 2)2^l$. The leftmost node in the range of a node from $C(j_2, l)$ is $(j_2 - 2)2^l + 1$. □

Lemma 4.12 *If $C^s(j_1, l)$ and $C^t(j_2, l)$ are clusters of the same level in two different layers L_s and L_t, and $s < t - 1$ then $j_1 \neq j_2$.*

Proof: Suppose $j_1 = j_2$. Then a node $v \in C^t(j_2, l)$ is in the range of every node from $C^s(j_1, l)$ and has to be in layer at most L_{s+1}. □

The *limit* f_i of layer L_i is defined as the source, for $i = 0$, and as $\max_{v \in L_{i-1}}\{v + r(v)\}$, for $i > 1$. An *active* leader is a leader of its cluster in whose range are some points larger than the limit of its layer. For every $p = 0, 1, 2, 3, 4$, we define the *stripe* S_p as the set of all active leaders for which $j(v)$ mod 5=p (in all layers). Lemma 4.11 implies

Lemma 4.13 *For every stripe S_p, every layer L, and every level l, there is at most one active leader.*

Lemma 4.14 *Let $u < v$ be active leaders in the same layer. Then v is in the range of u, and, if $l(v) > l(u)$ then also u is in the range of v.*

Proof: v is in the range of u because v is smaller than the limit of the layer. If $l(v) > l(u)$ then $r(v) \geq 2^{l(u)+1}$ but $v - u \leq r(v) \leq 2^{l(u)+1} - 1$. □

An active leader in layer L is *nonessential* if it is X-nonessential, where X is the set of all active leaders in this layer. Otherwise it is *essential.* In the sequel, "leader" means "essential active leader".

The definition of (essential) leaders and Lemma 4.14 imply

Lemma 4.15 *Let u and v be leaders in the same layer, such that $l(u) > l(v)$ or ($l(u) = l(v)$ and $u < v$). Then u is not in the range of v and $u + r(u) < v + r(v)$.*

The *main* leader in layer L is the leader $v \in L$ which maximizes $v + r(v)$. Lemma 4.15 implies that the main leader of every layer is on the lowest level and closest to the limit of the layer. Consider consecutive layers $L = L_i$ and $L^* = L_{i+1}$. Let m be the minimum level of a leader in L and M the maximum level of a leader in L^*.

Lemma 4.16 *If $M - m \geq 2$ then leaders of levels $\leq M - 2$ in L are in the range of the leader of level M in L^*.*

Proof: Let f be the limit of L. The leader of level M in L^* is larger than f but at most $f + 2^m$. The leader of level $M - 2$ in L is smaller than f but at least $f + 1 - (2^{M-2+1} - 1) = f - 2^{M-1} + 2$. The distance between those points is $2^m + 2^{M-1} - 2 \leq 2^{M-2} + 2^{M-1} - 2 < 2^M$ which does not exceed the range of the leader on level M in L^*. □

Lemma 4.17 *If $M - m \geq 3$ then $u - r(u) \leq v - r(v)$, where u is the leader of level M in L^* and v is a leader of level $\leq M - 3$ in L.*

Proof: Similar to that of Lemma 4.16. □

A leader $u \in L_i$ is *j-removable*, for $j \geq i$, if it is not a main leader and there exists a sequence $(a_0, ..., a_x)$, $0 \leq x \leq j - i$, such that a_k is the main leader in layer L_{i+k}, for $k = 0, 1, ..., x - 1$, and a_x is a leader in L_{i+x} (not necessarily main) with $a_x - r(a_x) \leq u$. Nodes $a_0, ..., a_x$ can inform u that it is not the main leader in its layer.

Lemma 4.18 *In every sequence of at most $\log R$ layers $L_i, L_{i+1}, ..., L_j$ there are at most $O(\log R)$ leaders which are not j-removable.*

Proof: For $k = i, i + 1, ..., j$, let l_k be the highest level on which there is a leader in L_k or there is a leader u in one of the layers $L_{k+1}, ..., L_j$ in whose range is the point $f_k + 1 - 2(2^{l+1} - 1)$, (i.e., to the right of layer L_k there is a leader reaching as far left as can reach a leader from layer L_k of level l.

Lemma 4.16 implies that all leaders of levels $\leq l_k - 2$ in layer L_{k-1}, except the main leader, are j-removable, for all $k = i + 1, ..., j$. Lemma 4.17 implies that $l_{k-1} \geq l_k - 3$. Hence the number of leaders from layer L_{k-1} which are not j-removable is 1, if $l_{k-1} = l_k - 3$ (the main leader only), and $1 + l_{k-1} - (l_k - 2)$, if $l_{k-1} \geq l_k - 2$. Hence this number does not exceed $4 + l_{k-1} - l_k$. Consequently, the total number of not j-removable leaders in layers $L_i, ..., L_j$ is at most $\sum_{k=i+1}^{j}(4 + l_{k-1} - l_k) + l_j = 4(j - i) + l_i \leq 5 \log R$. □

We are now ready to give a high-level description of a broadcasting protocol working in time $O(D + \log^2 R)$. It works in two stages.

Stage 1. Preprocessing
In every layer L_i find all leaders that are not $\min(i + \log R, D)$-removable and assign them consecutive numbers 1,2,..., starting from highest level to the lowest, on each level starting from the leftmost to the rightmost leader. (The main leader has the largest number in each layer.)

Stage 2. Proper broadcasting
Leaders in layer L_i broadcast the source message sequentially, in order from Stage 1. For $i > 1$, the ith iteration starts after getting the source message by leaders in L_i from the main leader of L_{i-1}.

Lemma 4.19 *The above algorithm is correct and works in time $O(T + D + \log R)$, where T is the time of preprocessing.*

Proof: Among nonremovable leaders in every layer, the main leader maximizes $v+r(v)$. This leader transmits the message to the next layer. Partition the sequence $(L_0, L_1, ..., L_D)$ of all layers into subsequences (L_0), $(L_1, ..., L_{\log R})$, $(L_{\log R+1}, ..., L_{2\log R})$, $(L_{y\log R+1}, ..., L_D)$, where $y = \lfloor D/\log R \rfloor$. In each subsequence $(L_{i\log R+1}, ..., L_{(i+1)\log R})$ the number of leaders that are not $(i + 1)\log R$-removable is $O(\log R)$. Hence the total running time of Stage 2 is $O(y \log R) = O(D + \log R)$. $\square$

We now show that Stage 1 can be executed in time $O(\log^2 R)$. It consists of the following 5 phases.

Phase 1. Find leaders of clusters in all layers.

Phase 2. Find limits of all layers and active leaders in each layer.

Phase 3. Remove nonessential leaders in all layers.

Phase 4. Remove all $\min(i + \log R, D)$-removable leaders in each layer.

Phase 5. In every layer, number all nonremovable leaders, starting from highest level to the lowest, on each level starting from the leftmost to the rightmost leader.

Phase 1 is completed using Algorithm 1 from Subsection 4.1, that finds a leader in a cluster in time $O(\log R)$. This algorithm works correctly if the work of nodes in a given cluster is not disturbed by nodes from other clusters. We have to find leaders in all clusters of all layers. To this end we find leaders in each level separately. In each level computations are divided into 10 subphases. Subphases 1,...,5 are devoted to work in layers of even indices and subphases 6,...,10 to work in layers of odd indices. In the pth subphase, $p = 0, 1, 2, 3, 4$, work is restricted to clusters $C(j, l)$ for which j mod 5=p. Lemmas 4.11 and 4.12 guarantee that two clusters working simultaneously do not disturb each other because nodes from these clusters are not in each other's range. This implies that Phase 1 can be completed in time $O(\log^2 R)$.

Phase 2 can be done simultaneously with phase 1. At the end of Algorithm 1 the elected leader v of a cluster sends the message $(v, r(v))$. By Lemma 4.11, all nodes in the range of v hear only v when it broadcasts this message. Hence all nodes from layer L_i, $i > 0$, can learn the limit f_i of this layer because among broadcasting leaders from L_{i-1} there is the main leader of this layer. (Recall that f_0 is the source.) After learning the limit, every leader knows if it is active. Moreover, all leaders from L_i can find the main leader from L_{i-1}. Phase 2 can be completed in time $O(\log^2 R)$.

In Phase 3 we remove nonessential leaders. Upon completion of phase 2 every leader knows if it is active. If an active leader v heard the message $(u, r(u))$ from an active leader in its layer (it can verify this since it knows the limit of its layer), then it considers itself nonessential if $u+r(u) > v+r(v)$ or $(u+r(u) = v+r(v)$ and $u < v)$. Phase 3 can be completed in time $O(\log^2 R)$.

Phase 4 consists of $\log R$ iterations. Before the ith iteration, nodes in every layer L_j know the position and range of the main leader from layer L_{j-i}. In the ith iteration, every leader from L_j broadcasts this message. If a leader from L_{j-i} gets this message and realizes that it is not the main leader (i.e., it gets a message $(u, r(u))$, where $u \leq f$ and $u + r(u) > f$, where f is the limit of L_{j-i}) , it considers itself as $\min(i + \log R, D)$-removable. Notice that, after phase 2, nodes in L_{i+1} know the range and position of the main leader from L_i. Phase 4 is executed in 10 subphases, similarly as phase 1, in order to avoid conflicts between leaders of different clusters. Phase 4 can be completed in time $O(\log^2 R)$.

Phase 5 can be done simultaneously with phase 4. A leader v from L_j that broadcasts parameters of the main leader from layer L_{j-i} adds to this message its own position and range. Every essential leader v in a layer knows all essential leaders of higher levels and smaller leaders of the same level. After getting a message about the main leader from its own layer, v can compute which of these leaders got the same message. All of them are removable. Hence upon completion of phase 4 every nonremovable leader knows the number of nonremovable leaders of levels higher or equal than its own layer. This number incremented by 1 is the consecutive number of this leader. Phase 5 can be completed in time $O(\log^2 R)$. Hence we get

Lemma 4.20 *Stage 1 (preprocessing) can be performed in time $O(\log^2 R)$.*

The above preprocessing works if all nodes in the configuration start their computations simultaneously. This assumption is very strong and can be avoided. We partition the broadcasting process into *supersteps*. Every superstep consists of broadcasting the source message in consecutive $\log^2 R$ layers. There are $S = \lceil D/\log^2 R \rceil$ supersteps. In the first superstep the message is broadcast to layers $L_1, ..., L_{\log^2 R}$, in the second superstep to layers $(L_{\log^2 R+1}, ..., L_{2\log^2 R})$, and so on. In each superstep we use the above described algorithm, hence total time is still $O(D + \log^2 R)$.

In order that the algorithm work correctly, all nodes of a superstep have to start computations simultaneously. Moreover, every node has to know to which layer of the superstep it belongs. This knowledge is acquired in the following way. All steps of the algorithm are numbered 0,1,2,... . For $i = 0, 1, 2$, we call *wave* W_i the set of steps whose numbers are congruent to i mod 3. Waves W_0 and W_1 will be used in determining supersteps and numbering layers. Wave W_2 will be used for broadcasting in supersteps. In this way knowledge of D is not necessary to accomplish broadcasting in prescribed time.

Let z be the maximum time to broadcast for configurations of depth $\log^2 R$. Supersteps are determined as follows. The source sends a message which is then propagated from layer to layer on wave W_0, i.e., all nodes that heard the message or noise, broadcast in the next time step of this wave. Then the source sends another (confirmation) message which is again propagated from layer to layer on wave W_1. However in each layer the propagation is delayed by 1 step. (Notice that most of the time nodes hear only noise.) Suppose that steps of each wave are renumbered by consecutive natural numbers. Then nodes from ith layer get the confirmation signal delayed by $i-1$ with respect to the first signal. In this way they can compute the number of their layer. Computations in superstep 1 start on wave W_2 in step $\log^2 R$. (At this point all nodes know the number of their layer.) Computations in superstep 2 start in step $z + log^2 R$, in superstep 3 they start in step $2z + \log^2 R$, and so on. Since $z = O(\log^2 R)$, this synchronization does not increase total time $O(D + \log^2 R)$.

In order to complete our argument we need to remove the technical assumption that all nodes are on one side of the source. (If all nodes are smaller than the source, the algorithm is similar to the above, where we assumed that all nodes are larger than the source.) Suppose that nodes are situated in arbitrary integer points of the line and s is the source. For every layer L_i we define two integers lf_i and rf_i called the *left* and *right* limit, respectively.

$lf_0 = rf_0 = s$, and for $i > 0$,
lf_i =min $\{v - r(v) : v \in L_{i-1} \cup L_{i-2} \cup ... \cup L_0\}$,
rf_i =max $\{v + r(v) : v \in L_{i-1} \cup L_{i-2} \cup ... \cup L_0\}$.

Consider two segments $Left_i = \{lf_i, lf_i + 1, ..., lf_{i-1} - 1\}$ and $Right_i = \{rf_{i-1} + 1, ..., rf_i\}$. If the left end is larger than the right end we consider the segment to be empty. The following lemma can be proved by induction on i.

Lemma 4.21 *For all $i > 0$, $L_i \cap (Left_i \cup Right_i) = L_i$ and $L_j \cap (Left_i \cup Right_i) = \emptyset$, for all $j \neq i$. If the segment $Left_i$ is nonempty, it lies to the left of the source, and if the segment $Right_i$ is nonempty, it lies to the right of the source.*

A node $v \in L_i$ is the *main sender to the right* if $v+r(v)$=max$\{u+r(u) : u \in L_i\}$ and $v+r(v) > rf_i$. If there are more than 1 such nodes, the main sender to the right is the one with largest range.

A node $v \in L_i \cap Left_i$ is the *left sender* if $v+r(v)$=max$\{u+r(u) : u \in L_i \cap Left_i\}$ and $v+r(v) > s$. (Recall that s denotes the source.) If there are more than 1 such nodes, the left sender is the one with largest range.

The main sender to the left and the right sender are defined symmetrically. Notice that the main sender to the right is the left sender if it is smaller than the source.

The left sender $v \in L_i$ is *important* if $v + r(v) > u + r(u)$ for all left senders u of preceding layers. If $v + r(v) \leq u + r(u)$ then v cannot inform any new nodes larger than the source (from the next layer). Similarly, the right sender $v \in L_i$ is *important* if $v - r(v) < u - r(u)$ for all right senders u of preceding layers.

Lemma 4.22 *The number of important senders is $O(\log R)$.*

Proof: We give the argument for left senders, right senders are analogous. It is enough to show that if u and v are important left senders in layers L_k and L_l, respectively, where $k < l - 1$, then $r(v) > 2r(u)$. Indeed, both u and v are smaller than the source and $v < u$ because u belongs to a layer with smaller index. $k < l - 1$ implies $v < u - r(u)$. Since v is important, $v + r(v) > u + r(u)$. Hence $r(v) > 2r(u)$. □

Important senders cannot be omitted because they may be main senders. On the other hand, if the main sender to the right (in a given layer) is larger than the source, and in the same layer there is an important left sender far away from it, these nodes cannot communicate and hence both of them have to broadcast. We must guarantee that they do not interfere in transmitting information to the right.

We now give a high-level description of a broadcasting protocol, for arbitrary configurations. As before, it consists of two stages.

Stage 1.
Phase 1 is as before. In phase 2 limits lf_i and rf_i and hence segments $Left_i$ and $Right_i$ are determined. Leaders elected in clusters broadcast their position and range, as well as their layer number. Using this information, all nodes in layer L_j can compute which node from layer L_{j-1} is the main sender (to the right or to the left). Moreover, if a node is a left or right sender, this information reaches the source which can compute important senders. If, in a given layer, the main sender to the right is smaller than the source, all nodes from this layer larger than the source are in its range. They learn that they are nonessential and do not participate in further computations. The situation is similar for main senders to the left.

The rest of phase 2 (finding active leaders) and phases 3, 4 and 5 are performed with no changes, separately for nodes larger than the source and separately (symmetrically) for nodes smaller than the source. As before, Stage 1 takes time $O(\log^2 R)$.

Stage 2.
The proper broadcasting stage is slightly modified. All steps are divided into 4 waves W_0, W_1, W_2, W_3, where W_i consists of steps whose numbers are congruent to i mod 4. Nodes smaller than the source broadcast to the left on wave W_0 and nodes larger than the source broadcast to the right on wave W_1. Both parts are executed by the previously described algorithm. Nonremovable nodes from $L_i \cap Left_i$ ($L_i \cap Right_i$) broadcast immediately after getting the source message from the main sender to the left (right).

The only problem is caused by important right (left) senders. If such a sender exists, it must broadcast the message because it may be the main sender. If, however, an important right (left) sender is not main, it can possibly interfere with the main sender in transmitting the message to the left (right). If these two nodes broadcasted simultaneously, some receivers would hear only noise. For this reason we reserve two additional waves: W_2 for left important senders and W_3 for right important senders. A node must know that it is an important sender to broadcast on the appropriate wave. To spread this knowledge the source attaches to the source message a list of all important senders. When the message reaches an important sender, it learns its status from the list and relays the message together with the list, on the appropriate wave.

It remains to estimate the running time of Stage 2. We divide all computations into *periods.* A period is a maximal sequence of consecutive steps in which waves W_2 or W_3 are not used. By Lemma 4.18, if the number x of layers informed in a given period is $\Omega(\log R)$ then the length of the period is $O(x)$. Hence Stage 2 takes time $O(D + \log^2 R)$. This yields the second main result of this section, now valid for arbitrary configurations.

Theorem 4.2 *Broadcasting in a configuration of depth D can be done in time $O(D + \log^2 R)$.*

5 Lower bound with known range

This section is devoted to establishing the lower bound $\Omega(\frac{\log^2 R}{\log\log R})$ on broadcasting time of any deterministic protocol in our main model, i.e., even when each node knows its own range. This lower bound will show that protocols from section 4 are asymptotically optimal for constant D and for $D = \Omega(\log^2 R)$, respectively. For simplicity we use a more informal style than that from section 3. It is not difficult, however, to reformulate the argument using Turing machines, states of nodes, etc. As before, and for the same reasons, we need to take a broadcast message of sufficiently high Kolmogorov complexity.

Theorem 5.1 *For any deterministic broadcast protocol there exists a configuration of depth 2 on which it requires $\Omega(\frac{\log^2 R}{\log\log R})$ steps.*

Proof: Nodes are situated at nonnegative integers. 0 is the source and its range is R, which is a power of 2. Let $h = \lfloor \frac{1}{3} \log R \rfloor$ and $p = 2^h$. Let $x_0, x_1, ..., x_{h-1}$ be a decreasing sequence of integers defined as follows:

$$x_i = R + 1 - (3 \cdot 2^i - 2) \cdot 2^h.$$

Notice that $x_i \geq 1$, for all $i = 0, 1, ..., h-1$.

For all $j = 0, 1, ..., h-1$, we define I_j as the segment $\{x_j, x_j + 1, ..., x_j + p - 1\}$. These segments are pairwise disjoint and a segment with higher index is to the left of a segment with lower index. Layer 1 is a subset of the union of these intervals. Denote $y_j = R + p - j$ and let $r(v) = y_j - v$, for all $v \in I_j$. Thus the range of every node in the first layer is at least p. Every pair of nodes in the same segment are in each other's range. Moreover, all nodes from segment I_j are in the range of nodes from segments I_k, for $k > j$ but not in the range of nodes from segments I_k, for $k < j$. Integers y_j form a descending sequence and y_j is in the range of nodes from $I_0, ..., I_j$ but not in the range of other nodes from layer 1.

The adversary will choose sets $C_j \subseteq I_j$ of nodes. Whenever C_j is nonempty, the adversary places a node in y_j and assigns $r(y_j) = 0$. Such a node must be informed but cannot inform any other node. The entire configuration consists of the source, of the union of sets C_j, for $j = 0, 1, ..., h-1$

and of the above nodes y_j. More precisely, layer 1 is equal to $C_0 \cup ... \cup C_{h-1}$ and layer 2 consists of corresponding nodes y_j. Hence $D = 2$.

Assume that all nodes in the first layer already know the source message. (This requires one step.) We will show that subsets C_j can be chosen in such a way that $\Omega(\frac{h^2}{\log h})$ steps are needed to inform all nodes of layer 2. Notice that the source is not in the range of any other node, hence it cannot modify its actions according to adversary decisions.

Let $t = \lfloor \frac{h^2}{\log h} \rfloor$ and let A be a broadcast protocol informing any configuration of the above type in at most t steps. A node $v \in C_j$ is called *solitaire* in C_j if, in some step of A, v is the only broadcasting node from C_j.

Lemma 5.1 *Every nonempty set C_j must contain a solitaire.*

Proof: Suppose that C_j is a nonempty set without a solitaire. The node y_j is only in the range of nodes from $C_0 \cup ... \cup C_j$. In case $C_0 \cup ... \cup C_{j-1} = \emptyset$, node y_j can be informed only by nodes in C_j. The absence of solitaire in this set implies that y_j can only hear noise or nothing. $\square$

Lemma 5.2 *Fix j. For some nonempty set C_j, the number x of steps in which at least two nodes from C_j broadcast, according to protocol A, prior to the first step in which a solitaire in C_j broadcasts, is at least $m = \lfloor dh/\log h \rfloor$, where d is a constant to be determined later.*

Proof: Suppose not. Let $X_{t,m}$ be the set of binary sequences of length t containing at most m terms 1. Then $|X_{t,m}| \leq 2^{2dh}$, for sufficiently large t.

For all binary sequences α of length at most t define the following sets $A(\alpha) \subseteq I_j$. $A(\epsilon) = I_j$. If $\alpha = \alpha' \bullet 1$ then $A(\alpha)$ is the set of those nodes from $A(\alpha')$ that broadcast in step $|\alpha|$, assuming that $C_j = A(\alpha')$. If $\alpha = \alpha' \bullet 0$ then $A(\alpha)$ is the set of those nodes from $A(\alpha')$ that do not broadcast in step $|\alpha|$, assuming that $C_j = A(\alpha')$.

The number of sequences corresponding to runs of A in which $x < m$ is at most 2^{2dh}. Hence for at least one sequence α, the set $A(\alpha)$ has size at least $p/2^{2dh}$. For $d = 1/4$ this size is at least $2^{h/2} > 1$. Hence, if $C_j = A(\alpha)$, all nodes in C_j get the same input. Since there are at least 2 of them, the solitaire cannot be chosen. $\square$

Now suppose that during the selection process of the solitaire in C_j some nodes from C_k, $k > j$, broadcast in steps $t_1, ..., t_s$. Is it possible to take advantage of these steps in order to reduce the number of remaining steps in which nodes in C_j broadcast? We will show that this is not the case, by constructing a set C_j with a stronger property than above:

Lemma 5.3 *The number of steps other than $t_1, ..., t_s$ in which at least two nodes from C_j broadcast, according to protocol A, prior to the first step in which a solitaire in C_j broadcasts, is at least m.*

Proof: Define sets $A^*(\alpha)$ similarly to sets $A(\alpha)$. $A^*(\epsilon) = I_j$. If $\alpha = \alpha' \bullet 1$ and $|\alpha| \in \{t_1, ..., t_s\}$ then $A^*(\alpha)$ has value $A^*(\alpha')$ if the number of nodes in $A^*(\alpha')$ that broadcast in step $|\alpha|$ is different from 1, and has value $A^*(\alpha') \setminus \{v\}$ if v is the only node from $A^*(\alpha')$ that broadcasts in step $|\alpha|$ (under assumption that $C_j = A^*(\alpha')$). If $\alpha = \alpha' \bullet 0$ the set $A^*(\alpha)$ is defined as $A(\alpha)$ before.

Let y be the number of steps excluding $t_1, ..., t_s$ in which at least two nodes from C_j broadcast, according to protocol A, prior to the first step in which a solitaire in C_j broadcasts. As before, the number z of sequences corresponding to runs of A in which $y < m$, is at most 2^{2dh}. However, in the present case, sets $A^*(\alpha)$ do not form a partition of I_j: Their union may be a proper subset of I_j. Nevertheless, for every sequence α, at most s nodes (corresponding to steps $t_1, ..., t_s$) were excluded. Hence $\sum_\alpha (|A^*(\alpha)| + s) \geq |I_j| = p = 2^h$, and consequently

$$\sum_\alpha |A^*(\alpha)| \geq 2^h - zs \geq 2^h - 2^{2dh}s \geq 2^h - 2^{2dh}t = 2^h - 2^{2dh}\lfloor h^2/\log h \rfloor \geq 2^h/2,$$

for sufficiently large h. Hence there exists a sequence α such that $A^*(\alpha)$ is of size at least 2. $\square$

In order to finish the proof, we show that at least $\lfloor \frac{dh}{\log h} \rfloor h$ steps are required to inform all nodes in layer 2. Consider the segment I_{h-1}. We have shown that there exists a subset $C_{h-1} \subseteq I_{h-1}$ for which at least $\lfloor \frac{dh}{\log h} \rfloor$ "noisy" steps are required before a solitaire is chosen. If the latest of these steps exceeds $\lfloor \frac{dh}{\log h} \rfloor h$ then the proof is finished. Otherwise, we consider the segment I_{h-2}. There exists a subset $C_{h-2} \subseteq I_{h-2}$ for which at least $\lfloor \frac{dh}{\log h} \rfloor$ *additional* "noisy" steps are required before a solitaire is chosen. If the latest of these steps exceeds $\lfloor \frac{dh}{\log h} \rfloor h$ then the proof is finished. Otherwise, we proceed to the construction of C_{h-3}, and so on. After h stages the number of steps required will become at least $\lfloor \frac{dh}{\log h} \rfloor h = \Omega(\frac{\log^2 R}{\log \log R})$. $\square$

6 Conclusion

In this paper we studied deterministic distributed algorithms for broadcasting in radio networks. We were mostly interested in the influence of knowledge available to nodes on the efficiency of broadcasting. We have shown two broadcasting protocols for radio networks on the line, assuming only local knowledge available to every node: its own position and range (and the maximum R of all ranges). One protocol works in time $O(D\frac{\log^2 R}{\log \log R})$ and the other in time $O(D + \log^2 R)$, where D is the depth of the radio network. Under the same model of knowledge, we also proved the lower bound $\Omega(\frac{\log^2 R}{\log \log R})$ valid for any radio network in the plane. (A much sharper lower bound was proved in the case when nodes do not even know their own range. On the other hand, full knowledge of the network easily yields $O(D)$ protocols.) Together with the trivial lower bound D, this shows that our protocols are asymptotically optimal for constant D and for $D = \Omega(\log^2 R)$, respectively. This yields the problem of finding a protocol asymptotically optimal for any network. This could be achieved, e.g., by improving preprocessing in the second protocol from $O(\log^2 R)$ to $O(\frac{\log^2 R}{\log \log R})$. We do not know if this is possible. An even more challenging problem would be to generalize our protocols for arbitrary networks in the plane. The lower bound $\Omega(\frac{\log^2 R}{\log\log R})$ seems weak in case of the plane and we would not expect broadcasting protocols for this more general setting to be as efficient as the ones presented in this paper.

References

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

[2] R. Bar-Yehuda, O. Goldreich, and A. Itai, On the time complexity of broadcast in radio networks: An exponential gap between determinism and randomization, Proc. 6th ACM Symposium on Principles of Distributed Computing (1987), 98 - 108.

[3] R. Bar-Yehuda, A. Israeli, and A. Itai, Multiple Communication in Multi-Hop Radio Networks, Proc. 8th ACM Symposium on Principles of Distributed Computing (1989), 329 - 338.

[4] I. Gaber and Y. Mansour, Broadcast in Radio Networks, Proc. 6th Ann. ACM-SIAM Symp. on Discrete Algorithms, SODA'95, 577-585.

[5] E. Kranakis, D. Krizanc and A. Pelc, Fault-Tolerant Broadcasting in Radio Networks, Proc. 6th European Symposium on Algorithms, ESA'98, to appear.

[6] E. Kushilevitz and Y. Mansour, An $\Omega(D \log n)$ Lower Bound for Broadcast in Radio Networks, Proc. 12th Ann. ACM Symp. on Principles of Distributed Computing (1993), 65-73.

[7] E. Kushilevitz and Y. Mansour, Computation in Noisy Radio Networks, Proc. 9th Ann. ACM-SIAM Symp. on Discrete Algorithms, SODA'98, 236-243.

[8] M. Li and P. Vitányi, Introduction to Kolmogorov Complexity and its Applications, Springer Verlag, 1993.

[9] K. Pahlavan and A. Levesque, Wireless Information Networks, Wiley-Interscience, New York, 1995.

[10] K. Ravishankar, and S. Singh, Broadcasting on $[0, L]$, Discrete Applied Mathematics 53 (1994), 299 - 319.

[11] A. Sen and M. L. Huson, A New Model for Scheduling Packet Radio Networks, Proc. 15th Annual Joint Conference of the IEEE Computer and Communication Societies (IEEE INFOCOM'96) (1996), 1116 - 1124.