SIAM J. Comput.
Vol. 33, No. 4, pp. 870–891

TIME OF DETERMINISTIC BROADCASTING IN RADIO NETWORKS WITH LOCAL KNOWLEDGE*

DARIUSZ R. KOWALSKI† AND ANDRZEJ PELC‡

Abstract. We consider broadcasting in radio networks, modeled as undirected graphs, whose nodes know only their own label and labels of their neighbors. In every step every node acts either as a *transmitter* or as a *receiver*. A node acting as a transmitter sends a message which can potentially reach all of its neighbors. A node acting as a receiver in a given step gets a message if and only if exactly one of its neighbors transmits in this step.

Bar-Yehuda, Goldreich, and Itai [*J. Comput. System Sci.*, 45 (1992), pp. 104–126] considered broadcasting in this model. They claimed a linear lower bound on the time of deterministic broadcasting in such radio networks of diameter 3. This claim turns out to be incorrect in this model (although it is valid in a more pessimistic model [R. Bar-Yehuda, O. Goldreich, and A. Itai, *Errata Regarding "On the time complexity of broadcast in radio networks: An exponential gap between determinism and randomization,"* http://www.wisdom.weizmann.ac.il/mathusers/oded/p_bgi.html, 2002]). We construct an algorithm that broadcasts in logarithmic time on all graphs from the Bar-Yehuda, Goldreich, and Itai paper (BGI). Moreover, we show how to broadcast in sublinear time on all n-node graphs of diameter $o(\log\log n)$. On the other hand, we construct a class of graphs of diameter 4, such that every broadcasting algorithm requires time $\Omega(\sqrt[4]{n})$ on these graphs. In view of the randomized algorithm from BGI, running in expected time $\mathcal{O}(D\log n + \log^2 n)$ on all n-node graphs of diameter D (cf. also a recent $\mathcal{O}(D\log(n/D) + \log^2 n)$-time algorithm from [D. Kowalski and A. Pelc, *Proceedings of the 22nd Annual ACM Symposium on Principles of Distributed Computing*, Boston, 2003, pp. 73–82; A. Czumaj and W. Rytter, *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, Cambridge, MA, 2003, pp. 492–501]), our lower bound gives the first correct proof of an exponential gap between determinism and randomization in the time of radio broadcasting, under the considered model of radio communication.

Key words. broadcasting, distributed, deterministic, radio network

AMS subject classifications. 68W15, 68Q25, 68Q17

DOI. 10.1137/S0097539702419339

1. Introduction.

A radio network is modeled as an undirected connected graph whose nodes are transmitter-receiver devices. An edge e between two nodes means that the transmitter of one end of e can reach the other end. Nodes send messages in synchronous *steps* (time slots), measured by a global clock which indicates the current step number. In every step every node acts either as a *transmitter* or as a *receiver*. A node acting as a transmitter sends a message which can potentially reach all of its neighbors. A node acting as a receiver in a given step gets a message if and only if exactly one of its neighbors transmits in this step. The message received in this case is the one that was transmitted. If at least two neighbors of u transmit simultaneously

*Received by the editors December 7, 2002; accepted for publication (in revised form) November 12, 2003; published electronically May 5, 2004. A preliminary version of this paper appeared in the Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2002), November 2002, Vancouver, Canada, under the title "Deterministic Broadcasting Time in Radio Networks of Unknown Topology."

http://www.siam.org/journals/sicomp/33-4/41933.html

†Instytut Informatyki, Uniwersytet Warszawski, Banacha 2, 02-097 Warszawa, Poland, and Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, Saarbrücken, 66123 Germany (darek@mimuw.edu.pl). This work was done in part during this author's stay at the Research Chair in Distributed Computing of the Université du Québec en Outaouais, as a postdoctoral fellow. This author's research was supported in part by KBN grant 4T11C04425.

‡Département d'informatique, Université du Québec en Outaouais, Hull, QC J8X 3X7, Canada (pelc@uqo.ca). This author's research was supported in part by NSERC grant OGP 0008136 and by the Research Chair in Distributed Computing of the Université du Québec en Outaouais.

in a given step, none of the messages is received by u in this step. In this case we say that a *collision* occurred at u. It is assumed that the effect at node u of more than one of its neighbors transmitting is the same as that of no neighbor transmitting (i.e., a node cannot distinguish a collision from silence). We assume that nodes know only their own label and the labels of their neighbors. Apart from that, the only a priori information on the network available to nodes is a polynomial upper bound on the number of nodes.

One of the fundamental tasks in network communication is *broadcasting.* Its goal is to transmit a message from one node of the network, called the *source*, to all other nodes. Remote nodes get the source message via intermediate nodes, along paths in the network. In this paper we concentrate on one of the most important and widely studied performance parameters of a broadcasting scheme, which is the total time—that is, the number of steps it uses to inform all the nodes of the network. We measure complexity in terms of the number of nodes in the network.

1.1. Our results. In a seminal paper [3], Bar-Yehuda, Goldreich, and Itai considered broadcasting in radio networks in the model described above. They claimed a linear lower bound on the time of deterministic broadcasting in such radio networks of diameter 3. This claim turns out to be incorrect, although it is valid in a more pessimistic model [4]. In fact, as pointed out in [4], the error is due to a gap between two models handling collisions in radio broadcasting: that from [3] and a more pessimistic but equally reasonable one. (See more comments on this point in subsection 1.2.) As discussed below, a lot of work on radio broadcasting has been done following [3], most of it modeling collisions as in [3].

Using the same model as in [3], we construct an algorithm that broadcasts in logarithmic time on all graphs from [3]. Moreover, we show how to broadcast in sublinear time on all n-node graphs of diameter $o(\log\log n)$. On the other hand, we construct a class of graphs of diameter 4, such that every broadcasting algorithm requires time $\Omega(\sqrt[4]{n})$ on one of these graphs. In view of the randomized algorithm from [3] running in expected time $\mathcal{O}(D\log n + \log^2 n)$ on all n-node graphs of diameter D (cf. also a recent $\mathcal{O}(D\log(n/D) + \log^2 n)$-time algorithm from [23, 15]), our lower bound gives the first correct proof of an exponential gap between determinism and randomization in the time of radio broadcasting, in the model from [3].

1.2. Related work. Most of the results concerning broadcasting in radio networks can be divided into two parts: those which assume complete knowledge of the topology of the network at all nodes, or equivalently, dealing with centralized broadcasting for a given network, and those assuming only limited knowledge of the network at all nodes and dealing with distributed broadcasting in arbitrary networks.

Deterministic centralized broadcasting assuming complete knowledge of the network was first considered in [10], where a $\mathcal{O}(D\log^2 n)$-time broadcasting algorithm was given for all n-node networks of diameter D. In [18], $\mathcal{O}(D + \log^5 n)$-time broadcasting was proposed. On the other hand, in [1] the authors proved the existence of a family of n-node networks of radius 2, for which any broadcast requires time $\Omega(\log^2 n)$.

The study of deterministic distributed broadcasting in radio networks whose nodes have only limited knowledge of the topology was initiated in [3]. The authors assumed that nodes know only their own label and the labels of their neighbors (and that collision at a node has the same effect as silence). Under this scenario, a simple $\mathcal{O}(n)$-time broadcasting algorithm based on depth-first search (DFS) follows from [2]. In [3] the authors constructed a class of n-node graphs of diameter 3, and claimed that every broadcasting algorithm requires time $\Omega(n)$ on one of these graphs. In [19] this claim was further strengthened to a lower bound of $n-1$ on broadcasting time required on one of these graphs. It follows from the present paper that both these claims (concerning lower bounds) are incorrect.

As pointed out in [4], the linear lower bound from [3] is valid in a more pessimistic model than that of [3]. Namely, one could assume that in the case of a collision at 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 the effect being left to the adversary. That is, either noise caused by many transmitting neighbors may be undistinguishable from background noise or else one of the competing neighbors may prevail, and it is impossible to predict which situation occurs for a given collision. For this model, which seems equally reasonable to that from [3], the argument and the linear lower bound of [3] are valid (cf. [4]). In fact, as explained in [4], the error in [3] is due to the gap between these two models.

Many authors [5, 7, 8, 9, 11, 12, 13, 14, 16, 26] studied deterministic distributed broadcasting in radio networks under the even weaker assumption that nodes know only their own label (but do not know the labels of their neighbors). In all these papers the collision issue was modeled as in [3]. In [11] the authors gave a broadcasting algorithm working in time $\mathcal{O}(n)$ for arbitrary n-node networks, assuming that nodes can transmit spontaneously before getting the source message. It was shown in [24] that if nodes know only their own label, the argument from [3] can be modified to prove a lower bound $\Omega(n)$ on broadcasting time for networks of radius 2. Thus the algorithm from [11] is optimal.

In [11, 12, 13, 14, 16, 26] the model of directed graphs was used. Increasingly faster broadcasting algorithms working on arbitrary n-node (directed) radio networks were constructed, culminating with the $\mathcal{O}(n\log^2 n)$-time algorithm from [13]. Recently, a $\mathcal{O}(n\log n\log D)$-time broadcasting algorithm was shown in [22] for n-node networks of radius D. This was further improved to $\mathcal{O}(n\log^2 D)$ in [15]. In [14] the authors showed a lower bound $\Omega(n\log D)$ on broadcasting time for n-node networks of radius D. On the other hand, in [5, 7, 8, 9, 12, 14] the problem was to find efficient broadcasting algorithms on radio networks of maximum in-degree Δ.

Finally, randomized broadcasting algorithms in radio networks were studied, e.g., in [3, 15, 25, 23]. For these algorithms no topological knowledge of the network was assumed. In [3] the authors showed a randomized broadcasting algorithm running in expected time $\mathcal{O}(D\log n+\log^2 n)$. A faster algorithm, running in expected time $\mathcal{O}(D\log(n/D)+\log^2 n)$ was presented in [23] (see also [15]). In [25] it was shown that for any randomized broadcasting algorithm (and parameters $D\le n$), there exists an n-node network of diameter D requiring expected time $\Omega(D\log(n/D))$. It should be noted that the lower bound $\Omega(\log^2 n)$ from [1], for some networks of radius 2, holds for randomized algorithms as well. This shows that the algorithm from [23] is optimal.

1.3. Organization of the paper. In section 2 we summarize the communication model (taken from [3]) and the terminology used in this paper. Section 3 is devoted to showing a logarithmic broadcasting algorithm for the class of networks for which a linear lower bound was claimed in [3]. In this section we first describe the novel Procedure Echo, which is later used for more complicated algorithms. In section 4 we describe and analyze a broadcasting algorithm working in sublinear time on all shallow networks. This indicates that if a linear lower bound on broadcasting time can at all be proved (for networks of sublinear diameter), then it requires the construction of quite complicated networks. Section 5 is devoted to the proof of the lower bound $\Omega(\sqrt[4]{n})$ on broadcasting time in networks of bounded diameter. Finally, section 6 contains concluding remarks and open problems.

2. Model and terminology.

We consider undirected graphs whose nodes have distinct labels belonging to the set $\{0, 1, \dots, r\}$, where r is polynomial in the number n of nodes. The parameter r is known to all nodes. In the lower bounds we assume that n itself is known to all nodes. A distinguished node with label 0 is called the *source*. We denote by D the *radius* of the graph, i.e., the distance from the source to the farthest node. (For undirected graphs, the diameter is of the order of the radius.) The jth *layer* of a graph is the set of nodes at distance j from the source. We adopt the communication model used by Bar-Yehuda, Goldreich, and Itai [3]. It is summarized in the following definition.

Definition 2.1 (see [3]). *A* broadcast protocol *for radio networks is a multiprocessor (multinode) protocol, the execution of which proceeds in steps (time-slots) (numbered* $0, 1, \dots$*) as follows:*

1. *In the initial step* 0*, only the source transmits a message called the* source message.
2. *In each step each node acts either as a transmitter or as a receiver (or is inactive).*
3. *A node receives a message in a specific step if and only if it acts as a receiver in this step and exactly one of its neighbors acts as a transmitter in that step. The message received in this case is the message transmitted by that neighbor.*
4. *The action of a node in a specific step is determined as a function of its initial input (which consists of its own label and the labels of its neighbors) and the (sequence of) messages that it has received in previous steps. All nodes have identical copies of the same program.*
5. *A node may act as a transmitter in a step* > 0 *only if it has received a message in a previous time-slot (there are no "spontaneous" transmissions of nodes other than the source in step* 0*).*
6. *The broadcast is completed at step* t *if all nodes have received the source message at one of the steps* $0, 1, \dots, t$.

As in [3], we assume that a node cannot distinguish whether more than one neighbor or no neighbor transmitted in a given step; i.e., we work in the model without collision detection.

3. Logarithmic broadcasting in *BGI* networks.

In [3] the following class of $(n+2)$-node networks was defined. Let S be a nonempty subset of $\{1, \dots, n\}$. The network G_S is a graph (of radius 2) whose nodes are labeled $0, 1, \dots, n+1$. The set of edges of G_S is $E = \{(0, i) : 1 \le i \le n\} \cup \{(i, n+1) : i \in S\}$. Node 0 is the source, and node $n+1$ (the only node in layer 2) is called the *sink*. We refer to all networks G_S as *BGI-n* networks (see Figure 3.1).

It was claimed in [3] that for any broadcast protocol there is a *BGI-n* network on which this protocol works in time $\Omega(n)$. However, assuming the model from [3], which is the same as that from the present paper, the proof in [3] contains the following flaw. *Predetermined* sets of nodes transmitting in consecutive steps are fixed, and then a network is constructed in which some node is not informed during any of these steps. However, during the broadcasting process, the source may potentially acquire some information, which it may pass to other nodes, thus modifying the sets of nodes transmitting in subsequent steps. So, in fact, under the considered model, the proof from [3] works only for oblivious algorithms, in which sets of nodes transmitting in a given step must be fixed in advance.

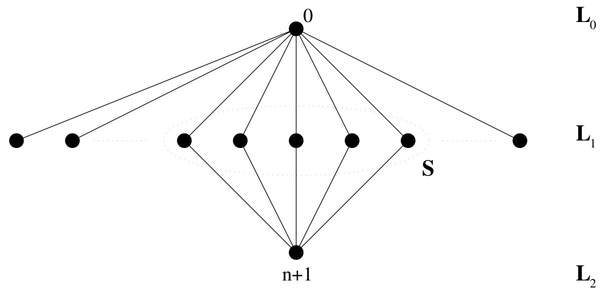

FIG. 3.1. *BGI-n network* G_S.

It turns out that not only is the argument from [3] erroneous under the considered model but, in fact, the above result itself is incorrect. (As mentioned in the introduction, the argument and the proof remain correct in a more pessimistic communication model.) Below we give an algorithm that broadcasts in all *BGI-n* networks in time $\mathcal{O}(\log n)$. The technique of selecting one out of many simultaneously transmitting neighbors, which is the main ingredient of this algorithm, will be further used in the construction of a much more involved algorithm which guarantees fast broadcasting in arbitrary networks of small radius.

The main idea of our algorithm is to simulate the collision detection capability in some nodes of the network. (Recall that collision detection is not available a priori in our model.) In order to get logarithmic broadcasting for *BGI-n* networks, it is enough to simulate collision detection at the source. To get sublinear broadcasting in all networks of radius $o(\log\log n)$, we will need to simulate this capability in many nodes from different levels.

Let A be a set of neighbors of the source (possibly unknown to it), and let $i \notin A$ be another neighbor of the source. Suppose that nodes in A want to transmit. Our goal is to let the source distinguish whether A has 0, 1, or more than 1 element. This can be done using the following 2-step procedure.

PROCEDURE ECHO (i, A).

Step 1. Every node in A transmits its label.

Step 2. Every node in $A \cup \{i\}$ transmits its label.

There are 3 possible effects of Procedure Echo (i, A) at the source.

Case 1. A message is received in Step 1 and no message in Step 2. In this case the source knows that A has 1 node and knows the label of this unique node.

Case 2. No message is received in Step 1 and a message (from i) is received in Step 2. In this case the source knows that A is empty.

Case 3. No message is received in either step. In this case the source knows that A has at least 2 nodes.

Procedure Echo is used to select one node in the set S of nodes connected to the sink, in a *BGI-n* network G_S. Once such a unique node is selected and transmits, the sink receives the source message, and broadcast is completed. This is done using the following algorithm. (In the original definition of *BGI-n* networks, nodes are labeled by consecutive numbers $0, 1, \ldots, n+1$, but we formulate our algorithm in the more general case, when labels are chosen arbitrarily from a set $\{0, 1, \ldots, r\}$, where r is polynomial in the number of nodes. Without loss of generality, assume that r is a power of 2.)

ALGORITHM BINARY-SELECTION-BROADCAST. In step 0, the source transmits the source message and the lowest label i of its neighbor (in the original definition of

BGI-n networks, $i = 1$). In step 1, node with label i transmits the source message and its degree. If this degree is 2 ($i \in S$), the sink receives the message and broadcast stops. Assume that the degree of i is 1.

All remaining steps $2, 3, \ldots$ are divided into segments of length 3. In the first step of each segment, the source transmits a range R of labels and orders the execution of Procedure Echo $(i, R \cap S)$ during the last two steps of the segment. (Notice that all nodes from layer 1 know if they are in $R \cap S$.) In the first segment, $R := \{1, \ldots, r/2\}$. If a range $R = \{x, \ldots, y\}$ is transmitted in a given segment, the range to transmit in the next segment is chosen according to the three possible effects of Procedure Echo $(i, R \cap S)$, described above. In Case 1, the sink is informed and broadcast stops. In Case 2, $R := \{y+1, \ldots, y+(y-x+1)/2\}$. In Case 3, $R := \{x, \ldots, (y+x-1)/2\}$. $\square$

THEOREM 3.1. *Algorithm Binary-Selection-Broadcast completes broadcasting in any BGI-n network in time $\mathcal{O}(\log n)$.*

Proof. Since the size of R transmitted in the ith segment is $r/2^i$, after at most $\log r \in \mathcal{O}(\log n)$ steps, the set $R \cap S$ contains exactly one node, and hence broadcast is completed in time $\mathcal{O}(\log n)$. $\square$

4. Sublinear time broadcasting in networks of radius $o(\log \log n)$.

In this section we construct a broadcasting algorithm working in time $o(n)$ on all n-node networks of radius $o(\log \log n)$. We will use the following results from the literature. The following two theorems assume that nodes know parameters r and d but do not assume any knowledge of the network topology.

THEOREM 4.1 (see [14]). *Consider a radio network modeled by an arbitrary graph (V, E), where V is a subset of $\{1, \ldots, r\}$. Let A and B be a partition of V such that all nodes in A have the same message m. Then there exists a protocol working in time $\mathcal{O}(\min(r, d \log(r/d)))$, which makes message m known to all nodes $v \in B$ having at least one and at most d neighbors in A.*

THEOREM 4.2 (see [17, 14]). *Given a radio network modeled by an arbitrary graph (V, E), where V is a subset of $\{1, \ldots, r\}$ and in which every node has a (possibly different) message, there exists a protocol working in time $\mathcal{O}(\min(r, d^2 \log r))$, upon the completion of which, every node of degree at most d learns the messages of all its neighbors.*

It should be mentioned that, while the protocols in the above theorems were obtained in a nonconstructive way, constructive counterparts of both these results (involving polynomial time local computations) are known and yield only slightly slower protocols. A constructive counterpart of Theorem 4.1 yielding a $\mathcal{O}(\min(r, d \cdot \text{polylog}\, r))$-time protocol follows from [20, 27], and a constructive counterpart of Theorem 4.2 yielding a $\mathcal{O}(\min(r, d^2 \log^2 r))$-time protocol follows from [21]. In our algorithm we use protocols from Theorems 4.1 and 4.2, but these constructive counterparts could be used as well, and our resulting broadcasting algorithm would still work in sublinear time.

The next result refers to radio networks of known topology.

THEOREM 4.3 (see [10]). *Consider a radio network modeled by an arbitrary graph (V, E), where V is a subset of $\{1, \ldots, r\}$, and assume that all nodes know the topology of the graph. Let A and B be a partition of V such that all nodes in A have the same message m. Then there exists a protocol working in time $\mathcal{O}(\log^2 r)$, which makes message m known to all nodes $v \in B$ having a neighbor in A.*

4.1. Broadcasting in networks of radius 2.

We first describe a sublinear time broadcasting algorithm working for all networks of radius 2. More generally, in every network G, this algorithm informs all nodes in levels 1 and 2 in sublinear time.

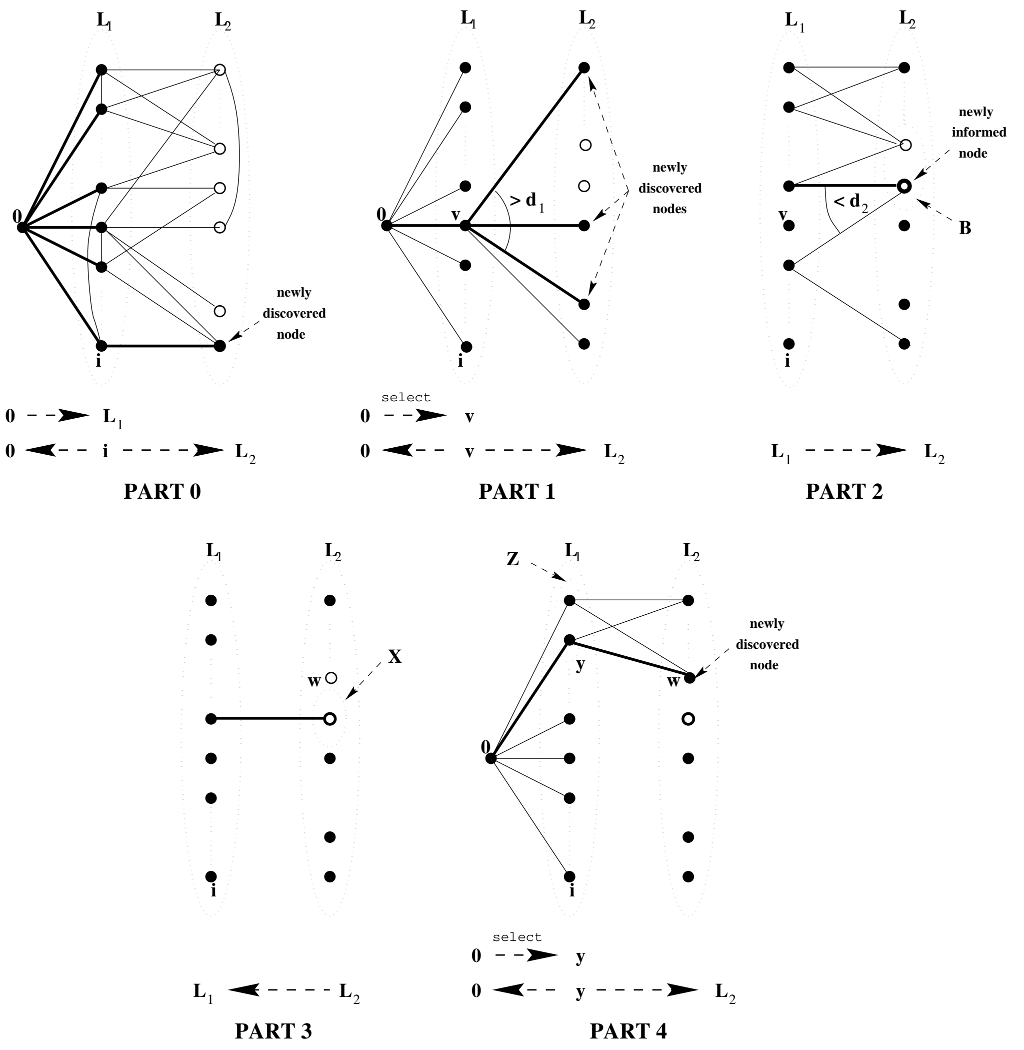

FIG. 4.1. *Algorithm $\mathcal{A}_2$.*

At each step of the execution, the source maintains a set DIS of *discovered* nodes: those about which it knows that they received the source message. The algorithm uses the polynomial upper bound r on the number of nodes, and two parameters d_1 and d_2, whose values will be specified later.

ALGORITHM $\mathcal{A}_2$ (see Figure 4.1).

Part 0. In step 0, the source transmits the source message, the set L_1, and the lowest label $i \in L_1$. In step 1, node with label i transmits the source message and the set C of its neighbors in L_2. The source sets the set DIS of discovered nodes to $\{0\} \cup L_1 \cup C$ and transmits it in step 2.

Part 1. Using a similar mechanism as in Algorithm Binary-Selection-Broadcast, the source selects a node in L_1 for which the number of undiscovered neighbors in L_2 is maximum. (Every node in L_1 can distinguish its neighbors in L_1 and in L_2.) In the next step the selected node transmits the source message and the set of its neighbors in L_2. The source adds them to discovered nodes and transmits the updated set DIS.

This process is repeated until there are no nodes in L_1 with more than d_1 undiscovered neighbors in L_2.

Part 2. Apply the protocol from Theorem 4.1 (with $d = d_2$) to the subgraph of G induced by $L_1 \cup B$ and to the partition (L_1, B), where B is the set of undiscovered nodes in L_2. This protocol makes the source message known to all undiscovered nodes from L_2 which have at most d_2 neighbors in L_1.

Part 3. Let X be the set of all nodes from L_2 which received the source message during Part 2 and have at most d_2 neighbors in L_1. Apply the protocol from Theorem 4.2 (for $d = d_1$) to the subgraph of G induced by $X \cup L_1$. The message transmitted by each node contains its label and the source message.

Part 4. All nodes from L_1 check if all their neighbors in L_2 got the source message. Nodes selected in Part 1 know that all their neighbors in L_2 were informed and discovered. Let Z be the set of nodes in L_1 which did not get a message in Part 3 from some of their undiscovered neighbors in L_2. Using a similar mechanism as in Algorithm Binary-Selection-Broadcast, the source selects one node in Z; then this node transmits (alone) the source message and the set of all its neighbors in L_2, and the source adds these neighbors to the set DIS of discovered nodes. After each selection, at least one currently undiscovered node gets the source message. This process continues until all nodes in L_1 know that all their neighbors in L_2 received the source message. □

THEOREM 4.4. *Algorithm $\mathcal{A}_2$ completes broadcasting in any n-node network of radius 2 in $\mathcal{O}(n^{2/3} \log n)$ time.*

Proof. Correctness is straightforward. The complexity of the algorithm is estimated as follows. Part 0 takes 3 steps. Each selection in Part 1 takes $\mathcal{O}(\log r)$ steps and there are at most n/d_1 selections; hence the entire Part 1 takes $\mathcal{O}((n/d_1) \cdot \log r)$ steps. In view of Theorem 4.1, Part 2 takes $\mathcal{O}(d_2 \cdot \log r)$ steps. In view of Theorem 4.2, Part 3 takes $\mathcal{O}(d_1^2 \log r)$ steps. Each selection in Part 4 takes $\mathcal{O}(\log r)$ steps and there are at most nd_1/d_2 selections; hence the entire Part 4 takes $\mathcal{O}((nd_1/d_2) \cdot \log r)$ steps. Taking $d_1 = \sqrt[3]{n}$ and $d_2 = \sqrt[3]{n^2}$, this adds up to $\mathcal{O}(n^{2/3} \log r) = \mathcal{O}(n^{2/3} \log n)$ steps. □

4.2. Extension to arbitrary networks of radius $o(\log \log n)$. We now describe an algorithm which broadcasts in sublinear time in arbitrary networks of radius $o(\log \log n)$. The algorithm uses the polynomial upper bound r on the number of nodes, and parameters d_j, d'_j, for $j = 2, 3, \ldots$, whose values will be specified later. The algorithm is constructed inductively. The construction is local, in the sense that every node constructs its part of the algorithm, using some coordination guaranteed by the properties and remarks listed below. After the kth phase of the algorithm, where k is an integer larger than 1, the following properties will be satisfied for some positive constant α.

Property 1. All nodes from layers L_j, $j = 1, \ldots, k$, know the source message and know to which layer they belong.

Property 2. For all $j = 1, \ldots, k$, a Procedure SEND(j), coordinated by the source, can be constructed, which has the following effect: if all nodes in L_{j-1} have the same message m and start at the same time, then all nodes in L_j learn message m. (SEND(1) consists of one step in which the source transmits.) Procedure SEND(j) lasts at most $\alpha \cdot (d_j + \log r) \log r$ time.

Remarks.

1. Let $j = 1, \ldots, k-1$, and $A \subseteq L_{j+1}$ be a set of nodes that want to transmit. It follows from Property 2 that a Procedure DETECT(j, A), coordinated by

the source, can be constructed, which enables every node in L_j to distinguish whether it has 0, 1, or more than 1 neighbor in A. Procedure DETECT(j, A) works as follows. Starting at a given time step t_0, all nodes in A repeat transmission in $1+\alpha\cdot(d_j+\log r)\log r$ consecutive time steps. Simultaneously, nodes in L_{j-1} perform Procedure SEND(j).
Each node $v \in L_j$ detects the number of its neighbors in A similarly as in Procedure Echo. By Property 2, v would get message m, during $\alpha\cdot(d_j+\log r)\log r$ steps after t_0, if nodes from A did not transmit. Hence v decides as follows. If it receives a message from a neighbor not in L_{j-1} during Procedure DETECT(j, A), it knows that it has exactly one neighbor in A and knows its label. Otherwise, two cases are possible. (1) If v receives only messages from L_{j-1} during Procedure DETECT(j, A), then it knows that none of its neighbors is in A. (2) If v receives no messages during Procedure DETECT(j, A), then it knows that at least two of its neighbors are in A.

2. Let $j = 0, \dots, k-1$, $i \in L_j$, and let $A \subseteq L_{j+1}$ be a subset of neighbors of i that want to transmit. A Procedure SELECT(j, i, A), coordinated by node i, can be constructed, in which node i selects one element in A. This can be done in $\alpha \cdot \log r$ steps, similarly as in Algorithm Binary-Selection-Broadcast, where node i plays the role of the source.

ALGORITHM SUBLINEAR-BROADCAST. In phase 1 do nothing. In phase 2 perform algorithm $\mathcal{A}_2$. Let $k > 1$. Suppose that Properties 1 and 2 are satisfied after phase k. We describe phase $k+1$. The source maintains a set DIS of *discovered* nodes: these are nodes from $L_{k-1} \cup L_k \cup L_{k+1}$ whose labels the source learned in phase $k+1$. At the beginning of phase $k+1$ this set is empty. Each node from L_k appends the number k of its layer to all its messages.
Description of phase $k+1$:

Part 0. The aim of this part is the verification of whether layer L_k is empty or not. If $L_k = \emptyset$, then the radius of the network is $D = k-1$ and broadcasting was completed at the end of phase $k-1$, by Property 1 for k. In this case the source sends a stop message. Otherwise, the source sends a message requesting the start of Part 1. Here is a detailed description of Part 0.

- The source initiates broadcast of the message "start verification in step t," by consecutive use of Procedures SEND(j), for $j = 1, \dots, k$, according to Property 2 for k. Step t is calculated to guarantee reception of this message by all nodes in L_j, for $j = 1, \dots, k$, i.e., to guarantee completion of all Procedures SEND(j).
- Verification of whether layer L_k is empty starts in step t.
  - Using Procedure DETECT$(k-1, L_k)$, nodes from L_{k-1} detect if they have neighbors in L_k.
  - Let A_{k-1} be the set of nodes from L_{k-1} which detected neighbors in L_k. Using Procedure DETECT$(k-2, A_{k-1})$, nodes from L_{k-2} detect if they have neighbors in A_{k-1}. This process continues with sets $A_i \subseteq L_i$, until the source detects if it has neighbors in A_1.
- If the source does not have neighbors in A_1 (i.e., A_1 is empty, which means that L_k is empty as well), then it initiates broadcast of the message "stop in step t_1" (for an appropriately calculated t_1), by consecutive use of Procedures SEND(j), for $j = 1, \dots, k-1$, according to Property 2 for k. Otherwise, a message requesting the start of Part 1 in an appropriately calculated step is sent similarly as above (to all layers L_j for $j = 1, \dots, k$).

Part 1. The aim of this part is selection of consecutive nodes from L_k which have at least d'_{k+1} undiscovered neighbors. This is done as follows, using a similar cascade of Procedures DETECT, as in Part 0.

- Let B_k be the set of nodes from L_k which have at least d'_{k+1} undiscovered neighbors. Using Procedure DETECT$(k-1, B_k)$, nodes from L_{k-1} detect if they have neighbors in B_k.
- Let B_{k-1} be the set of nodes from L_{k-1} which detected neighbors in B_k. Using Procedure DETECT$(k-2, B_{k-1})$, nodes from L_{k-2} detect if they have neighbors in B_{k-1}. This process continues with sets $B_i \subseteq L_i$, until the source detects if it has neighbors in B_1.
- If the source does not have neighbors in B_1 (i.e., B_1 is empty), then it initiates broadcast of the message "go to Part 2 in step t_2" (for an appropriately calculated t_2), by consecutive use of Procedures SEND(j) for $j = 1, \dots, k$, according to Property 2 for k. Otherwise the source selects one node from B_1, using Procedure SELECT$(0, 0, B_1)$.
- The selected node v performs SELECT$(1, v, B_2)$ to select one of its neighbors in B_2. This is continued until one node w in B_k is selected.
- Node w broadcasts a message (containing the source message, the label w, and labels of neighbors of w). All these neighbors get the source message. Moreover, broadcast is propagated along the path containing selected nodes from consecutive sets B_i. The source discovers neighbors of w and possibly w itself, updates the set DIS, and propagates this information to all nodes in layers $L_1, \dots, L_k$, using SEND procedures.

This selection process continues until all nodes in L_k have less than d'_{k+1} undiscovered neighbors.

Part 2. Let X be the set of all undiscovered nodes in L_k, and Y the set of all undiscovered nodes in $L_{k-1} \cup L_k \cup L_{k+1}$ which have at most d_{k+1} neighbors in X. A node can tell if it is in X, since in view of Property 1 it knows if it is in L_k, and after Part 1 of phase $k+1$ it knows whether it is in set DIS.

Apply the protocol from Theorem 4.2 (for $d = d_{k+1}$ and for the source message) to the subgraph of G induced by $X \cup Y$. At the end of Part 2, all nodes from Y got the source message.

Part 3. Consider the subgraph G of the radio network induced by the set of nodes V consisting of all undiscovered nodes in $L_{k-1} \cup L_k$, and of all undiscovered nodes in L_{k+1} which got the source message in Part 2. Each node knows if it is in V, in view of Part 2, of the knowledge of DIS gotten at the end of Part 1, and of Property 1.

Apply the protocol from Theorem 4.2 to the graph G (for $d = d'_{k+1}$). The message transmitted by each node contains its label and the source message. At the end of Part 3, all undiscovered nodes in L_k, which have at most d'_{k+1} neighbors in G, know which of their neighbors in G got the source message.

Part 4. All undiscovered nodes from L_k check if all their undiscovered neighbors got the source message. Consider the set Z of undiscovered nodes from L_k which did not get a message in Part 3 from some of their undiscovered neighbors. As in Part 1, we do the following:

- Procedures DETECT and SELECT are used to select one node in Z,
- this node transmits (alone) the source message and the set of all its neighbors in L_{k+1},
- this message is propagated to the source,
- the source updates the set DIS of discovered nodes (now DIS includes the selected node from Z and all of its neighbors) and propagates DIS to layer L_k.

After each selection, at least one currently undiscovered node in L_{k+1} gets the source message. This process continues until all nodes in L_k know that all their neighbors received the source message. $\square$

THEOREM 4.5. *Algorithm Sublinear-Broadcast completes broadcasting in arbitrary radio networks.*

Proof. It is enough to prove that Properties 1 and 2 are satisfied after each phase $k > 1$. First we prove them for $k = 2$, i.e., upon completion of Algorithm $\mathcal{A}_2$.

Property 1. By correctness of Algorithm $\mathcal{A}_2$, all nodes in L_1 and L_2 get the source message. All nodes in L_1 know that they are in L_1 because they got the message directly from the source. All nodes from L_2 got the message from some neighbor in L_1, by the description of Algorithm $\mathcal{A}_2$. On the other hand, they know that they are not in L_1, so they deduce that they are in L_2.

Property 2. Upon completion of Algorithm $\mathcal{A}_2$, all nodes in L_2 either are discovered or have at most d_2 neighbors in L_2. Procedure SEND(2) can be executed as follows. Broadcasting assuming knowledge of topology is executed in the graph containing all nodes from L_1 and all discovered nodes from L_2. This is done according to the protocol from Theorem 4.3 in time $\alpha_1 \cdot \log^2 r$. Broadcasting to the remaining nodes is done using the protocol from Theorem 4.1 in time $\alpha_2 \cdot (d_2 \log r)$. The property follows for $\alpha = \max(\alpha_1, \alpha_2)$.

Now assume that Properties 1 and 2 hold after phase k of Algorithm Sublinear-Broadcast. We have to prove that they hold after phase $k + 1$.

Property 1. Every node v in L_{k+1} gets a message from a neighbor w in L_k in phase $k + 1$. This is proved as follows. If v did not get a message until the end of Part 3, then all neighbors of v in L_k know that v did not get a message. Then at least one neighbor of v from L_k is selected in Part 4 and v gets a message from it. Node v learns that neighbor w is in L_k because w knows this by Property 1 after phase k and attaches this information to its message. Node v also knows that itself is not in L_i for $i \le k$, so it deduces that it is in L_{k+1}.

Property 2. Procedure SEND$(k + 1)$ is executed similarly as SEND(2) (described above), by replacing L_1 by L_k, L_2 by L_{k+1}, and d_2 by d_{k+1}. $\square$

Our next result estimates time complexity of Algorithm Sublinear-Broadcast for networks of small radius.

THEOREM 4.6. *Algorithm Sublinear-Broadcast completes broadcasting in time $o(n)$, for all n-node radio networks of radius $o(\log \log n)$.*

Proof. Fix a phase $k > 2$ of the algorithm. We estimate time complexity of each of its five parts separately.

Part 0. All Procedures SEND take a total of at most $2\alpha \cdot (\sum_{j<k}(d_j + \log r)\log r)$ steps. All Procedures DETECT take a total of at most $k - 1 + \alpha \cdot (\sum_{j<k-1}(d_j + \log r)\log r)$ steps. Hence the number of steps in the entire Part 0 is at most $\mathcal{O}(\sum_{j<k}(d_j + \log r)\log r) = \mathcal{O}(\sum_{j<k}(d_j + \log n)\log n)$.

Part 1. There can be at most (n/d'_k) selected nodes. We estimate the number of steps needed for each selection. All Procedures DETECT take a total of at most $k - 1 + \alpha \cdot (\sum_{j<k-1}(d_j + \log r)\log r)$ steps. All Procedures SELECT take a total of at most $\alpha(k - 1) \cdot \log r$ steps. Sending back a message to the source along a fixed path of selected nodes takes $k - 1$ steps. All Procedures SEND take a total of at most $\alpha \cdot (\sum_{j<k}(d_j + \log r)\log r)$ steps. Defining $\gamma = \alpha \cdot (\log r/\log n)^2$, we get the estimate

$$2\alpha \cdot \left(\sum_{j<k}(d_j + \log r)\log r\right) + \alpha(k-1)\cdot \log r + 2(k-1) \le 4\gamma \cdot \left(\sum_{j<k}(d_j + \log n)\log n\right)$$

on the number of steps for each selection. Hence the number of steps in the entire Part 1 is at most $4\gamma \cdot ((n/d'_k) \cdot \sum_{j<k}(d_j + \log n)\log n) \in \mathcal{O}((n/d'_k) \cdot \sum_{j<k}(d_j + \log n)\log n)$.

Part 2. By Theorem 4.2, this part takes $\mathcal{O}(d_k^2 \log r) = \mathcal{O}(d_k^2 \log n)$ steps.

Part 3. By Theorem 4.2, this part takes $\mathcal{O}((d'_k)^2 \log r) = \mathcal{O}((d'_k)^2 \log n)$ steps.

Part 4. Every node in set Z has at most d'_k undiscovered neighbors (otherwise it would be selected in Part 1). Every undiscovered neighbor w of a node $v \in Z$, from which v did not get a message in Part 3, is not in the graph G (defined in Part 3), hence it does not have the source message yet. By the description of Part 2, node w has more than d_k neighbors in L_{k-1}. Hence at most nd'_k/d_k selections of nodes in Z will be performed. The number of steps for each node is $\mathcal{O}(\sum_{j<k}(d_j + \log n)\log n)$, similarly as in Part 1. This gives a total of $\mathcal{O}((nd'_k/d_k)\sum_{j<k}(d_j + \log n)\log n)$.

We now choose the following parameters: $d_1 = \lfloor \log r \rfloor$, $d'_{i+1} = d_i^2$, and $d_{i+1} = (d'_{i+1})^2$. This gives the following estimates of the numbers of steps in different parts of phase k:

Part 0: $\mathcal{O}(d_{k-1}\log n) \subseteq \mathcal{O}(d_k^2 \log n)$.

Part 1: $\mathcal{O}((n/d'_k) \cdot d_{k-1}\log n) \subseteq \mathcal{O}((n\log n)/d_{k-1})$.

Part 2: $\mathcal{O}(d_k^2 \log n)$.

Part 3: $\mathcal{O}((d'_k)^2 \log n) \subseteq \mathcal{O}(d_k^2 \log n)$.

Part 4: $\mathcal{O}((nd'_k/d_k) \cdot d_{k-1}\log n) \subseteq \mathcal{O}((n\log n)/d_{k-1})$.

Thus the entire phase k lasts $\mathcal{O}((n/d_{k-1}) \cdot \log n + d_k^2 \log n)$ steps.

The last phase of the algorithm run on a radio network of radius D is $D+2$ (in fact only Part 0 of phase $D+2$ is executed). The total time of phases $k = 3, \ldots, D+2$ is at most $\mathcal{O}(d_{D+2}^2 \log n + (n\log n)/d_2) \subseteq \mathcal{O}(\min(n, (\log n)^{4^{D+3}+1} + n/\log^2 n))$. Since $D \in o(\log\log n)$, we get that this time is $o(n)$. By Theorem 4.4 the time of the first two phases (occupied by execution of Algorithm $\mathcal{A}_2$) is $\mathcal{O}(n^{2/3}\log n)$, which is $o(n)$ as well. Hence the entire Algorithm Sublinear-Broadcast completes broadcasting in time $o(n)$. ☐

5. Lower bound. In this section we show that for every deterministic broadcasting algorithm $\mathcal{A}$, there exists a network $G_{\mathcal{A}}$ of radius 2, with at most $2n$ nodes, on which $\mathcal{A}$ requires $\Omega(\sqrt[4]{n})$ steps to complete broadcast. This network is chosen from the family $\mathcal{C}_n$ of networks defined as follows. Every graph $G \in \mathcal{C}_n$ (see Figure 5.1) consists of the source 0 and two layers $L_1 = \{1, \ldots, n\}$ and $L_2 = \{n+1, \ldots, n+q\}$, where q is the largest odd integer smaller than $\sqrt[4]{n}$. The source is adjacent to all nodes in L_1, and every node in L_1 is adjacent to exactly one node in L_2. These are the only edges in G. If G is fixed, we denote by V the set $\{0\} \cup L_1 \cup L_2$ of all nodes of G. For every node v, we denote by N_v the set of its neighbors.

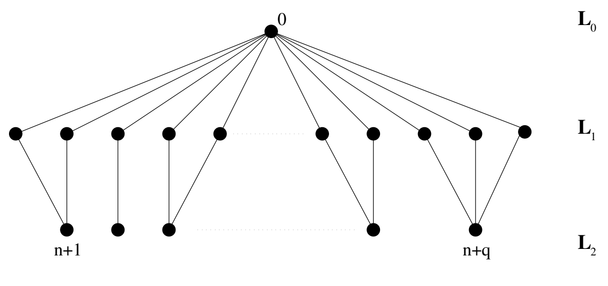

FIG. 5.1. *Network* $G \in \mathcal{C}_n$.

The idea of the proof is the following. We construct the network step by step, using consecutive steps of the fixed broadcasting algorithm $\mathcal{A}$, and *assuming* that particular nodes got particular messages in given steps. In order to express this, we use the notion of *abstract history* of a node, formally defined below. Intuitively, an abstract history of a node v at a given step k consists of a neighborhood of node v and of a sequence of messages received by this node until step k. Since the network is not yet constructed, neighborhoods of some nodes are not determined by step k, and consequently it is not yet known which abstract history will become the real one—the one given by algorithm $\mathcal{A}$ running on the final network. We can ensure that, if a given node had some abstract history up to a certain step, then it would behave in a given way (this is captured by the notion of abstract action function, defined below). Based on that we do the next step of the construction of the network (by determining neighborhoods of two nodes in layer L_2) and simultaneously define abstract histories of nodes in this step. These abstract histories are defined so as to prevent some nodes in layer L_2 of the network from getting any message for a long time. In particular, nodes of L_2 whose neighborhood is not yet determined have not gotten the source message so far.

When the construction is finished, we prove that if the algorithm $\mathcal{A}$ runs on the resulting network, then the real histories of all nodes are identical to the abstract (assumed) ones, and consequently some nodes of layer L_2 will indeed fail to receive the source message for $\Omega(\sqrt[4]{n})$ steps.

5.1. Construction. Fix a deterministic broadcasting algorithm $\mathcal{A}$. For this algorithm, running on any network, we define the following objects.

Histories and message format. H_k denotes the history of computation of algorithm $\mathcal{A}$ until the end of step k. This is the set $\{H_k(v) : v \in V\}$, where $H_k(v)$ is the history of computation at node v, until the end of step k. For any v and k, $H_k(v)$ is a sequence of (received) *messages* $(M_0(v), M_1(v), \ldots, M_k(v))$. Messages are defined inductively, as follows. $M_0(v)$ is either the triple $(\emptyset, \emptyset, \emptyset)$, called the *empty message*, or the triple $(0, N_0, \mathit{source\_message})$. $M_l(v)$ (for $l = 1, \ldots, k$) is the empty message if node v did not get any message in step l. Otherwise, it is a triple consisting of

- the label of node w from which node v received a message in step l,
- the set N_w,
- history $H_{l-1}(w)$.

Notice that we restrict attention to messages conveying the entire history of the transmitter. If a particular protocol requires transmitting specific information, the receiver can deduce this information from the received history, since programs of all nodes are the same. History $H_k(v)$ containing only empty messages is called the *empty history.*

Action function and sets of transmitters. Given algorithm $\mathcal{A}$, we denote by $\pi(v, N_v, H_{k-1}(v))$ the action of node v in step k, if its set of neighbors is N_v and its history until the end of step $k-1$ is $H_{k-1}(v)$. The values of the function π can be 1 or 0: if the value is 1, node v is sending the message $(v, N_v, H_{k-1}(v))$ in step k, otherwise it is receiving in step k. Under a fixed history H_{k-1}, we define the set of neighbors of v transmitting in step k as follows: $T_k(v) = \{w \in N_v : \pi(w, N_w, H_{k-1}(w)) = 1\}$.

We construct a network $G_{\mathcal{A}} \in \mathcal{C}_n$ on which $\mathcal{A}$ will work inefficiently. We first present the general overview of the construction and abstracts objects used in it. The construction is by induction on steps of the algorithm $\mathcal{A}$. The set of nodes $\{0\} \cup L_1 \cup L_2$, where $L_1 = \{1, \ldots, n\}$ and $L_2 = \{n+1, \ldots, n+q\}$, as well as all edges $\{(0, i) : i = 1, \ldots, n\}$ are given in the beginning. At each step of the induction some edges between nodes from L_1 and L_2 are added. Since at each stage of the construction only a part of the network $G_{\mathcal{A}}$ is specified, the new edges are constructed using algorithm $\mathcal{A}$ and some *abstract history* $\hat{H}_k$ until this step. Abstract histories at each node are parametrized with a nonempty set A representing a possible neighborhood of v to be constructed in a later step.

Abstract objects. Abstract objects (messages, histories, action function, transmitters) are abstract versions of real objects, used in the construction because real ones do not exist until the network is completely defined. Let $v \in V$ and $A \subseteq V$. An abstract history $\hat{H}_k(v, A)$ of node v, assuming that its neighborhood is A, is defined as a sequence $(\hat{M}_0(v, A), \hat{M}_1(v, A), \ldots, \hat{M}_k(v, A))$ of *abstract messages.* $\hat{M}_0(v, A) = M_0(v)$, and $\hat{M}_l(v, A)$, for $l > 0$, either is the empty message or is of the format $(w, B, \hat{H}_{l-1}(w, B))$, for some $w \in V$ and $B \subseteq V$. We will construct the abstract history step by step, in parallel with the construction of network $G_{\mathcal{A}}$. Notice that, in general, abstract histories and abstract messages are not necessarily linked to any particular protocol.

We also define the *abstract action function* $\hat{\pi}(v, A, \hat{H}_{k-1}(v, A))$ as an extension of the action function π described above. For every v and A, if $\pi(v, A, \hat{H}_{k-1}(v, A))$ is defined, then $\hat{\pi}(v, A, \hat{H}_{k-1}(v, A)) = \pi(v, A, \hat{H}_{k-1}(v, A))$. Otherwise, $\hat{\pi}(v, A, \hat{H}_{k-1}(v, A)) = 0$.

We now define sets of *abstract transmitters.* First consider a node v with neighborhood N_v fixed at the end of step k of the construction, and assume that neighborhoods N_w of all nodes $w \in N_v$ are also fixed. Under a fixed abstract history $\hat{H}_{k-1}$, we define the set of abstract transmitters $\hat{T}_k(v, N_v) = \{w \in N_v : \hat{\pi}(w, N_w, \hat{H}_{k-1}(w, N_w)) = 1\}$.

Now define sets of abstract transmitters for nodes whose neighborhood is not yet fixed. Suppose that S_k is the set of all nodes j in L_2 for which the neighborhood N_j is not fixed until the end of step k of the construction. Suppose that R_k is the set of nodes in L_1 that do not belong to any fixed neighborhood at the end of step k, i.e., $R_k = L_1 \setminus \bigcup_{j \in L_2 \setminus S_k} N_j$. (Additionally, let $S_0 = L_2$ and $R_0 = L_1$.) For nodes of R_k and S_k, we define the sets of abstract transmitters in step k as follows:

- if $v \in R_k$, then for any $j \in S_k$, $\hat{T}_k(v, \{0, j\}) = \{0\}$ if $\hat{\pi}(0, L_1, \hat{H}_{k-1}(0, L_1)) = 1$, and $\hat{T}_k(v, \{0, j\}) = \emptyset$ otherwise;
- if $v \in S_k$ and $R \subseteq L_1$, then $\hat{T}_k(v, R) = \{i \in R : \hat{\pi}(i, \{0, v\}, \hat{H}_{k-1}(i, \{0, v\})) = 1\}$.

Sets R_k and S_k will be defined dynamically in a formal way, during step k of the construction. We will prove that these formal definitions correspond to the meaning intended above for R_k and S_k, by proving Property 1 of the invariant after step k.

We now describe the inductive construction of the graph $G_{\mathcal{A}}$. We begin by defining the abstract history $\hat{H}_0$. $\hat{H}_0(v, A) = (\hat{M}_0(v, A))$, for all nodes v and sets A, where $\hat{M}_0(v, A)$ is the empty message for all $v \notin L_1$, and $\hat{M}_0(v, A) = (0, L_1, \textit{source\_message})$, for all $v \in L_1$.

We now begin step 1 of the construction, on the basis of step 1 of the algorithm and of the abstract history $\hat{H}_0$ already defined. To this end we will need the function First-Step-Selection, formally described below. We want to choose an element j of the set S (corresponding to L_2), to which the largest number of elements of the set R (corresponding to L_1) would transmit, if they were neighbors of j. Then we determine neighbors of j in L_1. When the function determines j, it also determines its neighborhood, and it deletes j from S. Hence, if S was the set of nodes with undetermined neighborhood before applying the function, it will preserve this property after applying it. When the neighborhood of j is determined, then (since neighbors of j are in L_1 and nodes in L_1 have exactly one neighbor in L_2) we automatically determine neighborhoods of neighbors of j. These neighbors are deleted from R, and hence R preserves the property of containing nodes with undetermined neighborhood, similar to S.

FUNCTION FIRST-STEP-SELECTION(R, S).

- Choose some node $j \in S$ such that the size of $X = \hat{T}_1(j, R)$ is maximal, and put two nodes from X to N_j (or one if X has one element, or nothing if X is empty); then remove these nodes from R. Remove j from S.
- Modify N_j as follows:
  if $N_j = \emptyset$, **then** put an arbitrary $i \in R$ to N_j and remove i from R;
  while there exists a node $i \in R$ such that $\hat{T}_1(v, R) = \{i\}$, for some $v \in S$
  do put i into N_j and remove i from R.
- Return (R, S, j, N_j).

Step 1 of the construction. The goal of step 1 of the construction is choosing two nodes j_1' and j_1 in L_2, together with their neighborhoods, in such a way that if some node from R_1 transmits in the first step of algorithm $\mathcal{A}$, then at least one other node from R_1 transmits as well. This is essential to guarantee the following property of abstract history $\hat{H}_1(0, L_1)$: no node from L_1 with yet undetermined neighborhood is heard by the source.

0. Initialize $R := L_1$ and $S := L_2$.
1. $(R, S, j_1', N_{j_1'}) :=$ FIRST-STEP-SELECTION(R, S);
   $R_1' := R$;
   $(R, S, j_1, N_{j_1}) :=$ FIRST-STEP-SELECTION(R, S);
   $R_1 := R$ and $S_1 := S$.
2. We construct the abstract history $\hat{H}_1$. Its definition corresponds to the definition of the "real" history, if neighborhoods are determined. Otherwise, the definition depends on the conditions on nodes and neighborhoods, the crucial case being the last item of the description below. History $\hat{H}_0$ is fixed; hence it is enough to define $\hat{M}_1(v, A)$, for all v, A. If $\hat{\pi}(v, A, \hat{H}_0(v, A)) = 1$, then $\hat{M}_1(v, A)$ is the empty message (transmitting nodes should not receive messages). Otherwise, $\hat{M}_1(v, A)$ is defined as follows:
   - Since nodes in $(L_1 \setminus R_1) \cup (L_2 \setminus S_1)$ have fixed neighborhoods, and also their neighbors have fixed neighborhoods, for each $v \in (L_1 \setminus R_1) \cup (L_2 \setminus S_1)$ we define $\hat{M}_1(v, N_v) = (w, N_w, \hat{H}_0(w, N_w))$, if $\hat{T}_1(v, N_v) = \{w\}$, and we define $\hat{M}_1(v, A)$ as the empty message in all other cases.
   - We define $\hat{M}_1(v, A)$ as the empty message, for all nodes $v \in S_1$ and all sets A.
   - We define $\hat{M}_1(0, L_1)$ as follows:
     – if $|\hat{T}_1(j_1', N_{j_1'}) \cup \hat{T}_1(j_1, N_{j_1})| \neq 1$, then $\hat{M}_1(0, L_1)$ is empty;
     – if $\hat{T}_1(j_1', N_{j_1'}) \cup \hat{T}_1(j_1, N_{j_1}) = \{i\}$, then $\hat{M}_1(0, L_1) = (i, N_i, \hat{H}_0(i, N_i))$. (Note that N_i is already defined at this point.)

     $\hat{M}_1(0, A)$ is defined as the empty message, for all $A \neq L_1$.
   - For every $v \in R_1$ and $j \in S_1$, if $\hat{T}_1(v, \{0, j\}) = \{0\}$, then $\hat{M}_1(v, \{0, j\}) = (0, L_1, \hat{H}_0(0, L_1))$, and $\hat{M}_1(v, A)$ is the empty message in all other cases.

This concludes the first step of the construction.

For any $k \geq 1$, the following invariant will be preserved after step k of the construction.

INVARIANT AFTER STEP k. The following objects are defined:
sets $S_l \subseteq L_2$ for $l = 0, 1, \ldots, k$;
sets $R_l \subseteq L_1$ for $l = 0, 1, \ldots, k$;

sets $R'_l \subseteq L_1$ for $l = 1, \dots, k$;
nodes j'_l, j_l such that $S_{l-1} \setminus S_l = \{j'_l, j_l\}$ and $R_{l-1} \setminus R_l = N_{j'_l} \cup N_{j_l}$ for $l = 1, \dots, k$.

The following properties hold:

1. Neighborhoods of nodes in $\{0\} \cup (L_1 \setminus R_k) \cup (L_2 \setminus S_k)$ are defined.
2. Histories $\hat{H}_k(v, A)$ are defined, for all nodes v and all sets A.
3. For all sets A, histories $\hat{H}_k(j, A)$ are empty for all $j \in S_k$, and histories $\hat{H}_{k-1}(j, A)$ are empty for all $j \in S_{k-1} \setminus S_k$.
4. For all nodes $j \in S_k \cup \{j_k\}$ and steps $l \leq k$, we have $|\hat{T}_l(j, R'_k)| \neq 1$.
5. For all nodes $j \in S_k$ and steps $l \leq k$, we have $|\hat{T}_l(j, R_k)| \neq 1$.
6. For all nodes $j \in S_{k-1} \setminus S_k$ and steps $l < k$, we have $|\hat{T}_l(j, N_j)| \neq 1$.

We now begin step $k + 1$ of the construction, on the basis of step $k + 1$ of the algorithm and of the invariant after step k. We will need a function similar to Function First-Step-Selection. Its aim is to choose $j \in S$ with the property as before (see the comment preceding the description of Function First-Step-Selection). This is done in the first item of the formal description given below. We also need to modify the neighborhood of j, so that choices (and elimination) of such nodes in previous steps of the construction do not yield a single transmitter to nodes with yet undetermined neighborhood. This is required in order to preserve Properties 4, 5, and 6 of the invariant. Modification of the neighborhood is done in the second item of the following formal description.

```
Function (k + 1)st-Step-Selection($R, S$).
  • Choose some node $j \in S$ such that the size of $X = \hat{T}_{k+1}(j, R)$ is maximal
    and put two nodes from $X$ to $N_j$ (or one if $X$ has one element, or nothing if
    $X$ is empty), then remove these nodes from $R$. Remove $j$ from $S$.
  • Modify $N_j$ as follows:
    if $N_j = \emptyset$, then put an arbitrary $i \in R$ to $N_j$ and remove $i$ from $R$;
      – set stop := 0,
      – while stop = 0 do
          * set stop := 1,
          * while there exists a node $i \in R$ such that $\hat{T}_l(j', R) = \{i\}$, for some
            $l \leq k + 1$, $j' \in S$
            do put $i$ into $N_j$ and remove $i$ from $R$,
          * while there exists a node $i \in N_j$ such that $\hat{T}_l(j, N_j) = \{i\}$, for some
            $l \leq k$
            do find another node $i' \in \hat{T}_l(j, R)$ (if it exists), put $i'$ into $N_j$, and
            remove $i'$ from $R$, set stop := 0;
  • Return $(R, S, j, N_j)$.
```

Step $(k + 1)$ of the construction (see Figure 5.2). The goal of step $(k + 1)$ of the construction is choosing two nodes, $j'_{k+1}, j_{k+1} \in S_k$ (in L_2), together with their neighborhoods (included in R_k), and defining abstract history $\hat{H}_{k+1}$, so as to satisfy the invariant after step $(k + 1)$ of the construction. Note that we do not initialize variables R and S because their values have been fixed after step k of the construction; indeed, at the beginning of step $k + 1$, we have $R = R_k$ and $S = S_k$.

1. $(R, S, j'_{k+1}, N_{j'_{k+1}}) := (k + 1)$st-Step-Selection$(R, S)$;
   $R'_{k+1} := R$;
   $(R, S, j_{k+1}, N_{j_{k+1}}) := (k + 1)$st-Step-Selection$(R, S)$;
   $R_{k+1} := R$ and $S_{k+1} := S$.
2. We construct the abstract history $\hat{H}_{k+1}$. Its definition corresponds to the definition of the "real" history, if neighborhoods are determined. Otherwise,

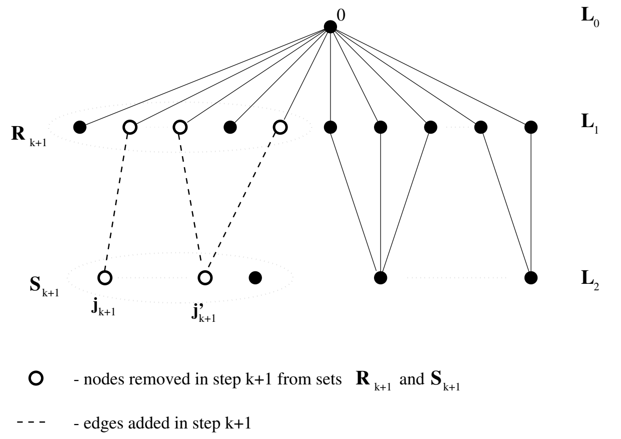
0
L0
L1
L2
R k+1
S k+1
j k+1
j' k+1
○ - nodes removed in step k+1 from sets R k+1 and S k+1
- - - - edges added in step k+1

FIG. 5.2. *Step $k+1$ of the construction of $G_{\mathcal{A}} \in \mathcal{C}_n$.*

the definition depends on the conditions on nodes and neighborhoods, the crucial case being the last item of the description below. History $\hat{H}_k$ is fixed; hence it is enough to define $\hat{M}_{k+1}(v, A)$, for all v, A. If $\hat{\pi}(v, A, \hat{H}_k(v, A)) = 1$, then $\hat{M}_{k+1}(v, A)$ is the empty message (transmitting nodes should not receive messages). Otherwise, $\hat{M}_{k+1}(v, A)$ is defined as follows:

- Since nodes in $(L_1 \backslash R_{k+1}) \cup (L_2 \backslash S_{k+1})$ have fixed neighborhoods, and also their neighbors have fixed neighborhoods, for each $v \in (L_1 \backslash R_{k+1}) \cup (L_2 \backslash S_{k+1})$ we define $\hat{M}_{k+1}(v, N_v) = (w, N_w, \hat{H}_k(w, N_w))$, if $\hat{T}_{k+1}(v, N_v) = \{w\}$, and we define $\hat{M}_{k+1}(v, A)$ as the empty message in all other cases.
- We define $\hat{M}_{k+1}(v, A)$ as the empty message, for all nodes $v \in S_{k+1}$ and all sets A.
- We define $\hat{M}_{k+1}(0, L_1)$ as follows:
  - if $|\bigcup_{j \in L_2 \backslash S_{k+1}} \hat{T}_{k+1}(j, N_j)| \neq 1$, then $\hat{M}_{k+1}(0, L_1)$ is empty;
  - if $\bigcup_{j \in L_2 \backslash S_{k+1}} \hat{T}_{k+1}(j, N_j) = \{i\}$, then $\hat{M}_{k+1}(0, L_1) = (i, N_i, \hat{H}_k(i, N_i))$. (Note that N_i is already defined at this point.)

  $\hat{M}_{k+1}(0, A)$ is defined as the empty message, for all $A \neq L_1$.
- For every $v \in R_{k+1}$ and $j \in S_{k+1}$, if $\hat{T}_{k+1}(v, \{0, j\}) = \{0\}$, then $\hat{M}_{k+1}(v, \{0, j\}) = (0, L_1, \hat{H}_k(0, L_1))$, and $\hat{M}_{k+1}(v, A)$ is the empty message in all other cases.

5.2. Analysis.

We first show that the invariant after step k of the construction holds if sets R_k, S_k are nonempty. This guarantees the correctness of the construction until one of these sets becomes empty, i.e., until all nodes either of L_1 or of L_2 have determined neighborhoods. Next we show that sets R_k, S_k are nonempty for $k \leq \frac{q-1}{2}$, where q is the largest odd integer smaller than $\sqrt[4]{n}$. This implies that the construction is correct until step $k = \frac{q-1}{2}$. Then we show how to finish the construction of network $G_{\mathcal{A}}$. Finally, we prove that histories determined by algorithm $\mathcal{A}$ running on network $G_{\mathcal{A}}$ are identical to the previously constructed abstract histories. In view of the invariant after step $k = \frac{q-1}{2} \in \Omega(\sqrt[4]{n})$ of the construction, this implies the desired lower bound on broadcasting time.

LEMMA 5.1. *The invariant after step k is preserved, for all $k \geq 1$ such that S_k and R_k are nonempty.*

Proof. The validity of the invariant after step 1 follows from the exit conditions in the first and second executions of function FIRST-STEP-SELECTION(R, S), in Part 1 of step 1 of the construction.

Assume that the invariant holds after step k and that S_{k+1} and R_{k+1} are nonempty. We prove that it holds after step $k+1$.

All required objects are defined by the construction in step $k+1$, using nonemptiness of S_{k+1} and R_{k+1}. It remains to prove the six properties.

1. This follows from Property 1 of the invariant after step k and from the construction of j'_{k+1}, j_{k+1}, and of their neighborhoods during Part 1 of the construction in step $k+1$.
2. $\hat{H}_{k+1}$ was defined in Part 2 of step $(k+1)$ of the construction.
3. The fact that $\hat{H}_{k+1}(v, A)$ is empty for all nodes $v \in S_{k+1}$ and all sets A follows from the assumption in Part 2 of step $(k+1)$ of the construction. The fact that $\hat{H}_k(j'_{k+1}, A)$ and $\hat{H}_k(j_{k+1}, A)$ are empty (j'_{k+1}, j_{k+1} are the only elements of $S_k \setminus S_{k+1}$) follows from Property 3 of the invariant after step k.
4. We prove that, for all nodes $j \in S_k \cup \{j_k\}$ and steps $l \leq k+1$, we have $|\hat{T}_l(j, R'_{k+1})| \neq 1$. This follows from the exit conditions of the external and of the first internal loop in function $(k+1)$ST-STEP-SELECTION(R, S) (more precisely, in the first execution of this function in Part 1 of step $(k+1)$ of the construction). The execution of the external loop ends if and only if the value of `stop` becomes 1, which means that the condition in the second internal loop is always false in the last turn of the external loop. Hence the condition of the first internal loop must be false at the end of the last turn of the external loop. This implies $|\hat{T}_l(j, R'_{k+1})| \neq 1$, for all nodes $j \in S_{k+1}$ and steps $l \leq k+1$.
5. We prove that for all nodes $j \in S_{k+1}$ and steps $l \leq k+1$ we have $|\hat{T}_l(j, R_{k+1})| \neq 1$. This follows by an argument similar as above, applied to function $(k+1)$ST-STEP-SELECTION(R, S) (now we refer to the second execution of this function in Part 1 of step $(k+1)$ of the construction).
6. The property $|\hat{T}_l(j, N_j)| \neq 1$, for all nodes $j \in S_k \setminus S_{k+1}$ and steps $l < k+1$, follows from the exit condition of the second internal loop in the first and second executions of function $(k+1)$ST-STEP-SELECTION(R, S), in Part 1 of step $(k+1)$ of the construction. Observe that the existence of i' in the second internal loop follows from Property 5 of the invariant after step k and from (the just proved) Property 4 of the invariant after step $k+1$. □

LEMMA 5.2. *The inductive construction of the network can be carried out for at least $\frac{q-1}{2}$ steps, where q is the largest odd integer smaller than $\sqrt[4]{n}$.*

Proof. Let $k \leq (q-1)/2$. Sets S_k are decreased by two nodes during one step; hence $S_k \neq \emptyset$, since $|S_0| = q$.

Claim. Sets R_k are decreased by at most $2q^3$ nodes during one step, at most q^3 for each of the chosen nodes j'_k, j_k.

This can be computed by analyzing loops in both executions of function kTH-STEP-SELECTION(R, S) in Part 1 of step k of the construction. Every turn of each of the internal loops increases the neighborhood $N_{j'_k}$ (resp., N_{j_k}) by at most one element and consequently decreases R by at most one element.

Consider the first execution. During the first internal loop, at most $kq \leq q^2/2$ nodes can be added to $N_{j'_k}$, since each action makes one set $\hat{T}_l(j, R)$ empty, where $l \leq k$, and $n+1 \leq j \leq n+q$. (Since we analyze subsequent executions of the loop in the function, symbol R in the expressions containing R corresponds to the current value of this variable, which changes dynamically. Hence values of these expressions may also change dynamically.)

During the second internal loop, at most $k-1 \le q/2$ nodes can be added to $N_{j'_k}$, since each action makes one set $\hat{T}_l(j'_k, N_{j'_k})$ of size at least 2, where $l \le k-1$. There may be at most $k-1 \le q/2$ executions of the external loop, since every execution of the external loop increases the size $|\hat{T}_l(j'_k, N_{j'_k})|$ to at least 2 for some $l \le k-1$ while performing the second internal loop. When $|\hat{T}_l(j'_k, N_{j'_k})| \ne 1$, for all $l \le k-1$, the execution of the external loop stops. Hence $N_{j'_k}$ is bounded by $(q^2/2+q/2)\cdot(q/2) \le q^3$, and consequently R is decreased at the rate of at most q^3 nodes per step. The same is true for the second execution. Hence R_k is smaller than R_{k-1} by at most $2q^3$ nodes, which concludes the proof of the claim.

Since $q < \sqrt[4]{n}$, we have $R_k \ne \emptyset$ for all $k \le (q-1)/2$. It follows that the construction can be carried out for $(q-1)/2$ steps. ◻

Using Lemma 5.2, the construction of the network $G_{\mathcal{A}}$ can be now concluded as follows. All nodes in $R_{(q-1)/2}$ are made adjacent to the only node in $S_{(q-1)/2}$. It follows from the construction that $G_{\mathcal{A}}$ belongs to the class $\mathcal{C}_n$ defined in the beginning of this section.

The histories $\hat{H}_k$ in consecutive steps of the construction were abstract histories defined in order to continue the construction in subsequent steps. The next lemma shows that the actual histories $H_k(v)$ of all nodes v of network $G_{\mathcal{A}}$ obtained by running algorithm $\mathcal{A}$ on this network, are identical to abstract histories $\hat{H}_k(v, N_v)$.

LEMMA 5.3. *Let $k \le (q-1)/2$ be a step of the execution of algorithm $\mathcal{A}$ on network $G_{\mathcal{A}}$. Then $H_k(v) = \hat{H}_k(v, N_v)$, for all nodes v of network $G_{\mathcal{A}}$.*

Proof. In the first step of the algorithm execution, the source transmits and all nodes in L_1 receive the message. Nodes in L_2 receive nothing. Hence $H_0(v) = \hat{H}_0(v, N_v)$ by definition of $\hat{H}_0$.

Note that the definition of abstract history in step 1 of the construction is the same as that in step $k+1$, taken for $k=0$. Hence it is not necessary to separately analyze step 1, and we can proceed with the argument by induction, for an arbitrary k.

Assume by induction that $H_k(v) = \hat{H}_k(v, N_v)$, where $k < (q-1)/2$. We prove $H_{k+1}(v) = \hat{H}_{k+1}(v, N_v)$ by showing $M_{k+1}(v) = \hat{M}_{k+1}(v, N_v)$. (Since $k+1 \le (q-1)/2$, the abstract history $\hat{H}_{k+1}$ is well defined, in view of Lemma 5.2.) Observe that, since $\hat{\pi}$ is an extension of π and $H_k(v) = \hat{H}_k(v, N_v)$, we have $\hat{\pi}(v, N_v, \hat{H}_k(v, N_v)) = \pi(v, N_v, H_k(v))$. Hence, if $\hat{\pi}(v, N_v, \hat{H}_k(v, N_v)) = 1$, then v acts as a transmitter in step $k+1$, and hence both $M_{k+1}(v)$ and $\hat{M}_{k+1}(v, N_v)$ are empty messages. Thus we assume in the following that $\hat{\pi}(v, N_v, \hat{H}_k(v, N_v)) = 0$, i.e., that v acts as a receiver.

Case 1. $v \in (L_1 \setminus R_{k+1}) \cup (L_2 \setminus S_{k+1})$.

By Property 1 of the invariant after step $k+1$ of the construction, v has a fixed neighborhood and all of its neighbors w have fixed neighborhoods. Since $\hat{H}_k(w, N_w) = H_k(w)$, we get $M_{k+1}(v) = \hat{M}_{k+1}(v, N_v)$.

Case 2. $v \in S_{k+1}$.

$\hat{M}_{k+1}(v, N_v)$ is the empty message by Property 3 of the construction invariant after step $k+1$. Let k' be the step in which the neighborhood N_v was constructed. Since $v \in S_{k+1}$, we have $k' > k+1$. By Property 6 of the invariant after step k' of the construction, $|\hat{T}_l(v, N_v)| \ne 1$, for all steps $l < k'$. In particular, $|\hat{T}_{k+1}(v, N_v)| \ne 1$. Since $\hat{H}_k(w, N_w) = H_k(w)$ for all $w \in N_v$, we have $T_{k+1}(v) = \hat{T}_{k+1}(v, N_v)$, and hence $T_{k+1}(v)$ is not a singleton. It follows that $M_{k+1}(v)$ is the empty message.

Case 3. $v = 0$.

If $j \in L_2 \setminus S_{k+1}$, then $\hat{T}_{k+1}(j, N_j) = T_{k+1}(j)$, since $\hat{H}_k(w, N_w) = H_k(w)$, for all $w \in N_j$. Hence $\bigcup_{j \in L_2 \setminus S_{k+1}} \hat{T}_{k+1}(j, N_j) = \bigcup_{j \in L_2 \setminus S_{k+1}} T_{k+1}(j)$. We consider three cases.

If $|\bigcup_{j \in L_2 \setminus S_{k+1}} \hat{T}_{k+1}(j, N_j)| > 1$, then $\hat{M}_{k+1}(0, L_1)$ is empty. In this case, $|T_{k+1}(0)| = |\bigcup_{j \in L_2} T_{k+1}(j)| > 1$ and hence $M_{k+1}(0)$ is the empty message.

If $\bigcup_{j \in L_2 \setminus S_{k+1}} \hat{T}_{k+1}(j, N_j) = \{i\}$, then $\hat{M}_{k+1}(0, L_1) = (i, N_i, \hat{H}_k(i, N_i))$. In this case, $\bigcup_{j \in L_2 \setminus S_{k+1}} T_{k+1}(j) = \{i\}$. By construction of j_{k+1}, we have $\hat{T}_{k+1}(j_{k+1}, R_{k+1}) = \emptyset$, and hence $\hat{T}_{k+1}(j, R_{k+1}) = \emptyset$, for all $j \in S_{k+1}$. Consequently, $T_{k+1}(j) = \hat{T}_{k+1}(j, N_j) \subseteq \hat{T}_{k+1}(j, R_{k+1}) = \emptyset$. It follows that $M_{k+1}(0) = (i, N_i, H_k(i))$. In view of $H_k(i) = \hat{H}_k(i, N_i)$, we get $M_{k+1}(0) = \hat{M}_{k+1}(0, L_1)$.

If $\bigcup_{j \in L_2 \setminus S_{k+1}} \hat{T}_{k+1}(j, N_j) = \emptyset$, then $\hat{M}_{k+1}(0, L_1)$ is the empty message. In this case, $\bigcup_{j \in L_2 \setminus S_{k+1}} T_{k+1}(j) = \emptyset$. The same reasoning as above gives $\hat{T}_{k+1}(j, R_{k+1}) = \emptyset$ for all $j \in S_{k+1}$. Consequently, $M_{k+1}(0)$ is the empty message.

Case 4. $v \in R_{k+1}$.

Since for all $j \in S_{k+1}$, $\hat{H}_{k+1}(j, N_j) = H_{k+1}(j)$ is the empty history, it follows that each node $v \in R_{k+1}$ can receive a message in step $k+1$ of $\mathcal{A}$ only from node 0, if this node transmits. If node 0 transmits in step $k+1$ of $\mathcal{A}$, then v receives the message $M_{k+1}(v) = (0, L_1, H_k(0))$. Since $H_k(0) = \hat{H}_k(0, L_1)$, by definition we have $\hat{T}_{k+1}(v, N_v) = \{0\}$. By construction of message $\hat{M}_{k+1}(v, N_v)$ we get $\hat{M}_{k+1}(v, N_v) = (0, L_1, \hat{H}_k(0, L_1))$. Since $H_k(0) = \hat{H}_k(0, L_1)$, we have $\hat{M}_{k+1}(v, N_v) = M_{k+1}(v)$. If node 0 does not transmit in step $k+1$ of $\mathcal{A}$, then $M_{k+1}(v)$ is empty. Since $H_k(0) = \hat{H}_k(0, L_1)$, by definition we have $\hat{T}_{k+1}(v, N_v) = \emptyset$. By construction, $\hat{M}_{k+1}(v, N_v)$ is the empty message. $\square$

THEOREM 5.4. *For any deterministic broadcasting algorithm $\mathcal{A}$, there exists a network $G_{\mathcal{A}}$ of radius* 2, *with at most* $2n$ *nodes, for which this algorithm requires time* $\Omega(\sqrt[4]{n})$.

Proof. Network $G_{\mathcal{A}}$ constructed above has $n + 1 + q \le 2n$ nodes, since q is the largest odd integer smaller than $\sqrt[4]{n}$. It has radius 2 by construction. Let $k = (q-1)/2$. By Lemma 5.2, S_k is nonempty. By Lemma 5.1 and Property 3 of the invariant after step k, histories $\hat{H}_k(j, N_j)$ are empty for all $j \in S_k$. By Lemma 5.3, histories $H_k(j)$ are empty for all $j \in S_k$. Hence no node in S_k receives the source message by step k of algorithm $\mathcal{A}$. It follows that algorithm $\mathcal{A}$ requires time $\Omega(\sqrt[4]{n})$ to broadcast on network $G_{\mathcal{A}}$. $\square$

Using the above technique we can prove the following more general result.

COROLLARY 5.5. *For any deterministic broadcasting algorithm $\mathcal{A}$ and any parameters $D \le n$, there exists an n-node network of radius D, for which this algorithm requires time* $\Omega(\sqrt[4]{nD^3})$.

6. Conclusion.

In this paper we studied deterministic broadcasting time in radio networks whose nodes know only their immediate neighborhood. We presented an algorithm for broadcasting in sublinear time in all networks of radius $o(\log\log n)$ and we proved a lower bound $\Omega(\sqrt[4]{n})$ on broadcasting time even in networks of radius 2. In view of the randomized algorithm from [3] running in expected time $\mathcal{O}(D \log n + \log^2 n)$ on all n-node graphs of diameter D, our lower bound proves an exponential gap between time of deterministic and randomized broadcasting in radio networks.

The main problem that remains open is the following. Is there a deterministic broadcasting algorithm running in sublinear time on all networks with sublinear radius, if nodes know only their immediate neighborhood? If complete knowledge of the network is available, the positive answer follows from [18].

REFERENCES

[1] N. Alon, A. Bar-Noy, N. Linial, and D. Peleg, *A lower bound for radio broadcast*, J. Comput. System Sci., 43 (1991), pp. 290–298.

[2] B. Awerbuch, *A new distributed depth-first-search algorithm*, Inform. Process. Lett., 20 (1985), pp. 147–150.

[3] R. Bar-Yehuda, O. Goldreich, and A. Itai, *On the time complexity of broadcast in multi-hop radio networks: An exponential gap between determinism and randomization*, J. Comput. System Sci., 45 (1992), pp. 104–126.

[4] R. Bar-Yehuda, O. Goldreich, and A. Itai, *Errata Regarding "On the time complexity of broadcast in radio networks: An exponential gap between determinism and randomization,"* http://www.wisdom.weizmann.ac.il/mathusers/oded/p_bgi.html, 2002.

[5] S. Basagni, D. Bruschi, and I. Chlamtac, *A mobility-transparent deterministic broadcast mechanism for ad hoc networks*, IEEE/ACM Trans. on Networking, 7 (1999), pp. 799–807.

[6] S. Basagni, A. D. Myers, and V. R. Syrotiuk, *Mobility-independent flooding for real-time multimedia applications in ad hoc networks*, in Proceedings of the IEEE Emerging Technologies Symposium on Wireless Communications & Systems, Richardson, TX, 1999.

[7] D. Bruschi and M. Del Pinto, *Lower bounds for the broadcast problem in mobile radio networks*, Distr. Comp., 10 (1997), pp. 129–135.

[8] I. Chlamtac and A. Faragó, *Making transmission schedule immune to topology changes in multi-hop packet radio networks*, IEEE/ACM Trans. on Networking, 2 (1994), pp. 23–29.

[9] I. Chlamtac, A. Faragó, and H. Zhang, *Time-spread multiple access (TSMA) protocols for multihop mobile radio networks*, IEEE/ACM Trans. on Networking, 5 (1997), pp. 804–812.

[10] I. Chlamtac and O. Weinstein, *The wave expansion approach to broadcasting in multihop radio networks*, IEEE Trans. on Communications, 39 (1991), pp. 426–433.

[11] B. S. Chlebus, L. Gasieniec, A. Gibbons, A. Pelc, and W. Rytter, *Deterministic broadcasting in unknown radio networks*, Distributed Computing, 15 (2002), pp. 27–38.

[12] B. S. Chlebus, L. Gąsieniec, A. Östlin, and J. M. Robson, *Deterministic radio broadcasting*, in Proceedings of the 27th International Colloquium on Automata, Languages and Programming (ICALP'2000), Lecture Notes in Comput. Sci. 1853, Springer-Verlag, Berlin, 2000, pp. 717–728.

[13] M. Chrobak, L. Gąsieniec, and W. Rytter, *Fast broadcasting and gossiping in radio networks*, in Proceedings of the 41st Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press, Redondo Beach, CA, 2000, pp. 575–581.

[14] A. E. F. Clementi, A. Monti, and R. Silvestri, *Selective families, superimposed codes, and broadcasting on unknown radio networks*, in Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, Philadelphia, 2001, pp. 709–718.

[15] A. Czumaj and W. Rytter, *Broadcasting algorithms in radio networks with unknown topology*, in Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science, Cambridge, MA, 2003, pp. 492–501.

[16] G. De Marco and A. Pelc, *Faster broadcasting in unknown radio networks*, Inform. Process. Lett., 79 (2001), pp. 53–56.

[17] P. Erdős, P. Frankl, and Z. Füredi, *Families of finite sets in which no set is covered by the union of r others*, Israel J. Math., 51 (1985), pp. 79–89.

[18] I. Gaber and Y. Mansour, *Broadcast in radio networks*, in Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, Philadelphia, 1995, pp. 577–585.

[19] F. K. Hwang, *The time complexity of deterministic broadcast radio networks*, Discrete Appl. Math., 60 (1995), pp. 219–222.

[20] P. Indyk, *Explicit constructions of selectors and related combinatorial structures, with applications*, in Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, Philadelphia, 2002, pp. 697–704.

[21] W. H. Kautz and R. R. C. Singleton, *Nonrandom binary superimposed codes*, IEEE Trans. on Inform. Theory, 10 (1964), pp. 363–377.

[22] D. Kowalski and A. Pelc, *Faster deterministic broadcasting in ad hoc radio networks*, in Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science, Berlin, Lecture Notes in Comput. Sci. 2607, Springer-Verlag, Berlin, 2003, pp. 109–120.

[23] D. Kowalski and A. Pelc, *Broadcasting in undirected ad hoc radio networks*, in Proceedings of the 22nd Annual ACM Symposium on Principles of Distributed Computing, Boston, 2003, pp. 73–82.

[24] D. Kowalski and A. Pelc, *Time of radio broadcasting: Adaptiveness vs. obliviousness and randomization vs. determinism*, in Proceedings of the 10th Colloquium on Structural Information and Communication Complexity, Umea, Sweden, 2003, pp. 195–210.

[25] E. Kushilevitz and Y. Mansour, *An $\Omega(D \log(N/D))$ lower bound for broadcast in radio networks*, SIAM J. Comput., 27 (1998), pp. 702–712.

[26] D. Peleg, *Deterministic Radio Broadcast with No Topological Knowledge*, manuscript, 2000.

[27] A. Ta-Shma, C. Umans, and D. Zuckerman, *Loss-less condensers, unbalanced expanders, and extractors*, in Proceedings of the 33rd Annual ACM Symposium on Theory of Computing, 2001, pp. 143–152.