Token Traversal in Ad Hoc Wireless Networks via Implicit Carrier Sensing

Tomasz Jurdzinski(✉), Michal Rozanski, and Grzegorz Stachowiak

Institute of Computer Science, University of Wrocław, Wrocław, Poland
tju@cs.uni.wroc.pl

Abstract. Communication problems in ad hoc wireless networks have been already widely studied under the SINR model, but a vast majority of results concern networks with constraints on connectivity, so called *strongly-connected networks*. What happens if the network is not strongly-connected, e.g., it contains some long but still viable "shortcut links" connecting transmission boundaries? Even a single broadcast in such ad hoc *weakly-connected* networks with uniform transmission powers requires $\Omega(n)$ communication rounds, where n is the number of nodes in the network. The best up-to-date (randomized) distributed algorithm, designed by Daum et al. [10], accomplishes broadcast task in $O(n \log^2 n)$. In this work we show a novel deterministic distributed implementation of token traversal in the SINR model with uniform transmission powers and no restriction on connectivity. We show that it is efficient even in a very harsh model of weakly-connected networks without GPS, carrier sensing and other helping features. We apply this method to span a traversal tree and accomplish broadcast in $O(n \log N)$ communication rounds, deterministically, provided nodes are equipped with unique IDs in the range $[1, N]$ for $N \geq n$. This result implies an $O(n \log n)$-round randomized solution that does not require IDs, which improves the result from [10]. The lower bound $\Omega(n \log N)$ for deterministic algorithms proved in our work shows that our result is tight without randomization. Our implementation of token traversal routine is based on a novel implicit algorithmic carrier sensing method and a new type of selectors, which might be of independent interest.

Keywords: Wireless ad hoc networks · SINR · Token traversal
Broadcast · Deterministic and randomized algorithms
Algorithmic carrier sensing · Selectors · BTD trees

1 Introduction

We study distributed algorithms in ad hoc wireless networks in the SINR model with uniform transmission powers. We consider an *ad hoc* setting, where both

The work of the first and the third author was supported by the Polish National Science Centre grant DEC-2012/07/B/ST6/01534 and the work of the second author was supported by the Polish National Science Centre grant 2014/13/N/ST6/01850.

S. Das and S. Tixeuil (Eds.): SIROCCO 2017, LNCS 10641, pp. 15–33, 2017.
https://doi.org/10.1007/978-3-319-72050-0_2

capability and knowledge of nodes are limited — nodes know only the basic parameters of the SINR model (i.e., $\alpha, \beta, \mathcal{N}$, $\mathcal{P}$, to be defined later). We assume that each node knows its distinct ID and the range of IDs $[N] = \{1, \ldots, N\}$. Such setting appears in networks without predefined infrastructure of base stations, access points, etc. It reflects various real scenarios, such as: large sets of sensors distributed in an area of rescue operation, environment monitoring, or prospective internet of things applications.

Token traversal. We focus on the problem of token traversal, in which a software-defined token needs to visit all (or a subset of) nodes in the network. More precisely, in the beginning there is a distinguished node, called a source, which has a status of the token owner. In each round only one node can have a status of the token owner. The ownership of the token can be passed to a neighbor via a message; in wireless network, however, it can be challenging to select an unvisited neighbor to which the token can be passed, due to ad hoc structure and interferences. The token traversal is accomplished if every participating node has been a token owner for at least one round. Token traversal is a fundamental task in distributed system, and a tool of building algorithms to solve more complex communication and computation tasks.

Broadcast problem. The broadcast problem was extensively studied in the model of graph-based radio networks over the years, while distributed algorithms for the SINR model have been presented only in recent years. However, all these solutions were either randomized, or relied on the assumption that nodes of a network know their own coordinates in a given metric space (GPS), or used carrier sensing capabilities or the advantage of power control (ability to change transmission power).

Challenges and our approach. Almost all communication algorithms analyzed in the SINR model assumed *strong connectivity* of a network. That is, connectivity of a network is guaranteed by links (u, v) such that efficient transmission from u to v (and from v to u) is possible provided interference at v caused by other nodes of a network is limited by some fixed constant. Our aim is to provide solutions which work in the most harsh and general scenario, when connectivity might rely on weak links and thus allow for efficient transmissions only in the case of no other (or at least very small number of) transmitters in the whole network; it is called a *weakly-connectivity* model, and subsumes the strong-connectivity one. Moreover, we assume that communicating devices have very limited capabilities, in particular, they do not use randomization, availability of locations, carrier sensing, or power control. The key challenge in design of algorithms for the model considered in this paper is the assumption that nodes of a network have initially no information about network topology. The fact that nodes use a single wireless channel and therefore their messages might collide is an additional obstacle for efficient communication.

Our results. We present a deterministic algorithm that traverses a token along any (even weakly-connected) wireless ad hoc network, under the uniform-power

SINR, in amortized $O(\log N)$ rounds. More specifically, the token is propagated along a specific spanning tree, called a BTD (Breadth-Then-Depth) tree in time proportional to the number of participants multiplied by $O(\log N)$. It can be applied to perform broadcast in weak connectivity ad hoc networks in $O(n \log N)$ communication rounds, and is supported by a corresponding lower bound. Our result implies $O(n \log n)$ randomized algorithm with high probability (i.e., with probability polynomially close to 1), even if IDs are not available (see Sect. 7), which improves the $O(n \log^2 n)$ algorithm of Daum et al. [10].

We also introduce new tools, which might be applicable in different scenarios and problems. Firstly, inspired by Echo procedure that simulates collision detection in radio networks [33], we introduce a kind of implicit carrier sensing allowing fast testing of emptiness of sets. Secondly, in order to efficiently select nodes from dense areas of a network, we introduce a new combinatorial structure called a *witnessed strong selector*.

Related work. The SINR model was extensively studied recently, both from the perspective of its structural properties [17,25,26] and design of algorithms [10,13–15,18,19,21,24,27,36,37]. First wave of algorithmic research on communication under SINR constraints focused on local problems. This includes in particular the local broadcast and link scheduling [13,16,28,29,37].

Token-based algorithms were considered in related models of multiple-access channel and radio networks, e.g., [3,31]. In radio networks, an $O(\log N)$ procedure of token passing was presented in [6,34], and combined with the BTD tree traversal. In the SINR model of weak devices subsumed by and less complex than the weak-connectivity model, efficient implementation of a token was provided [31].

A few deterministic solutions are known for the broadcast problem, most of them use information about location of nodes and assume strong connectivity. Broadcast can be accomplished deterministically in time $O(D \log^2 n)$ in such setting [23,24], where D is the diameter of the communication graph. The randomized results on broadcast in ad hoc settings include [10,22]. Solutions with complexity, respectively $O((D \log n) \log^{\alpha+1} g)$ and $O(D \log^2 n)$ are presented for strong connectivity networks, where g is a parameter depending on the geometry of the network. Recently Halldorsson et al. [14] proposed an algorithm which can be faster assuming that nodes are equipped with some extra capabilities.

For weak connectivity networks Daum et al. have provided $\Omega(n)$ lower bound for broadcast, even in 2-broadcastable networks. They also showed that the problem can be solved in $O(n \log^2 n)$ time with high probability.

In the related multi-hop radio network model on symmetric networks, the broadcast problem is well examined [1,2,9,11,12,30,32,35].

Due to limited space, we defer some proofs to the full version of the paper.

2 The Network Model

We consider a wireless single-channel network consisting of nodes located on the 2-dimensional Euclidean plane, where interferences are modeled according to

SINR (*Signal-to-Interference-and-Noise Ratio*) constraints. The model is determined by fixed parameters: path loss $\alpha > 2$, threshold $\beta > 1$, ambient noise $\mathcal{N} > 0$ and transmission power $\mathcal{P}$. Given nodes u, v and a set of concurrently transmitting nodes $\mathcal{T}$, the value of $SINR(v, u, \mathcal{T})$ is defined as

$$SINR(v, u, \mathcal{T}) = \frac{\mathcal{P} \cdot d(v, u)^{-\alpha}}{\mathcal{N} + \sum_{w \in \mathcal{T} \setminus \{v\}} \mathcal{P} \cdot d(w, u)^{-\alpha}} \tag{1}$$

where $d(x, y)$ denotes the distance between locations of x and y.

A node u successfully receives a message from v iff $v \in \mathcal{T}$ and $SINR(v, u, \mathcal{T}) \geq \beta$, where $\mathcal{T}$ is the set of nodes transmitting at the same time. *Transmission range* is the maximal distance at which a node can be heard provided there are no other transmitters in the network. Without loss of generality we assume that the transmission range is equal to 1. This assumption implies that the relationship $\mathcal{P} = \mathcal{N}\beta$ holds. However, it does not affect generality and asymptotic complexity of presented results.

Communication graph. The *communication graph* $G = (V, E)$ of a given network consists of all nodes from V and edges $\{v, u\}$ between nodes that are within distance of at most 1, i.e., $\{v, u\} \in E$ iff $d(u, v) \leq 1$. The communication graph, defined as above, is a *weak connectivity graph* [10,20].

Synchronization and content of messages. We assume that algorithms work synchronously in rounds. In a single round, a node can transmit or receive a message from other node in the network and perform local computation. A message transmitted by a node in a round might contain the original broadcast message and additional information of size $O(\log N)$.

Knowledge of nodes. Each node has a unique identifier from the set $[N]$, where $N > n$ and n is the number of nodes in the network. The value of n or its polynomial approximation is known to the nodes. Moreover, nodes know the range of IDs space N, and the SINR parameters – $\mathcal{P}, \alpha, \beta, \mathcal{N}$.

Considered problems. We consider a general *token traversal* problem defined in Sect. 1. A node $v \neq s$ starts participating in an execution of an algorithm only after receiving the first message from another node. (This is so-called *non-spontaneous wake-up* model.) We also consider the *broadcast* problem which is to deliver a message from the designated source node s to all the nodes in the network, perhaps through relay nodes as not all nodes are within transmission range of the source in multi-hop networks.

Complexity measure. Time (or round) complexity of an algorithm is the number of rounds after which an execution of an algorithm is finished. We assume worst-case complexity measure.

Constructive vs non-constructive solutions. We say that an algorithm is *constructive* if the algorithm for a given value of N can be built in time polynomial with respect to N. The algorithms delivered in this work are constructive, because of the fact that our algorithms use combinatorial structures only for the range of parameters guarantying polynomial time construction.

3 Preliminaries and Combinatorial Tools

The set of integers $\{1, 2, \ldots, n\}$ is denoted by $[n]$ and $\{i, i+1, \ldots j\}$ by $[i, j]$.

A *transmission schedule* is defined by a sequence $\mathcal{S} = (S_1, ..., S_t)$ of subsets of $[N]$, where the ith set determines nodes transmitting in the ith round of the schedule. That is, a node with $\mathrm{ID}(v) \in [N]$ transmits in round i of an execution of $\mathcal{S}$ if and only if $v \in S_i$.

In the following, V denotes the set of nodes of a network on the plane. Thus, each node $v \in V$ is determined by its identifier $\mathrm{ID}(v)$ in $[N]$ and its coordinates on the plane. In descriptions of algorithms, $\mathrm{ID}(v)$ is sometimes identified with v. Let $\mathcal{B}(x, r)$ denote the ball of radius r around point x on the plane. We identify $\mathcal{B}(x, r)$ with the set of nodes of the network that are located inside this ball on the plane. For a node $v \in V$, $N_v = \{w \in V \mid d(v, w) \leq 1\}$ denotes the set of neighbors of v in the communication graph. For $a > b > 0$, $\chi(a, b)$ denotes the largest possible size of a set of points X included in a ball of radius b such that $d(x, y) > a$ for each distinct $x, y \in X$.

A node w is in the *graph distance* i from v if i is the length of a shortest path connecting w and v in the communication graph. Assume that a distinguished source node $s \in V$ is fixed. Then, $L_i \subseteq V$ denotes the set of nodes in graph distance i from s (layer i). Thus, e.g., $L_0 = \{s\}$ and $L_1 = N_s$.

We say that a node v *awakes* w in an execution of an algorithm if the first message successfully received by w is sent by v.

3.1 Combinatorial Tools

In this section we introduce combinatorial tools applied in our token traversal algorithm. A set $S \subseteq [N]$ *selects* $x \in X$ from $X \subseteq [N]$ when $S \cap X = \{x\}$. A sequence $\mathcal{S} = (S_1, \ldots, S_t)$ of sets over $[N]$ is called (N, k)-strongly selective family (or (N, k)-ssf) if for each subset $X \subseteq [N]$ such that $|X| \leq k$, and each $x \in X$ there is $i \in [t]$ such that S_i selects x from X.

Lemma 1 [8]. *There exists a (N, k)-ssf of size $O(\min\{k^2 \log(N/k), N\})$ for each $k \leq N$.*

Now, we introduce the notion of a *witnessed strong selector*, which is a generalization of strongly selective families.

Witnessed strong selector. A sequence $\mathcal{S} = (S_1, \ldots, S_m)$ of sets over $[N]$ satisfies *witnessed strong selection property for a set* $X \subseteq [N]$, if for each $x \in X$ and each $y \notin X$ there is a set $S_i \in \mathcal{S}$ such that $X \cap S_i = \{x\}$ and $y \in S_i$. A sequence $\mathcal{S} = (S_1, \ldots, S_m)$ is a (N, k)-*witnessed strong selector* (or (N, k)-wss) of size m if for every subset $X \subseteq [N]$ of size k the family $\mathcal{S}$ satisfies the witnessed strong selection property for X.

Note that any (N, k)-wss is also, by definition, an (N, k)-ssf. Additionally, (N, k)-wss guarantees that each element outside of a given set X of size k has to be a "witness" of selection of every element from X. Below we state an upper bound on the optimal size of (N, k)-wss.

Lemma 2. *For each positive integers N and $k \leq N$, there exists an (N, k)-wss of size $O(k^3 \log N)$.*

Construction of witnessed strong selectors. We aim at the efficient algorithm constructing a (N, k)-wss for a constant k. Our solution is inspired by the algorithm of Clementi et al. [7], which employs the technique of conditional probabilities.

Lemma 3. *For each integers $0 < k < N$, a (N, k)-wss of size $O(k^3 \log N)$ can be constructed in time $N^{O(k)}$; in particular, it can be constructed in polynomial time for any $k = O(1)$.*

3.2 SINR Related Properties

We say that distinct nodes $u, v \in A$ form a *closest pair of nodes* (u, v) in the set A if $d(u, v) = min_{x,y \in A, x \neq y}\{d(x, y)\} \leq 1/2$.[1]

Below, we state the fact that u can hear v if (u, v) is a closest pair, v is transmitting and there is no other transmitter in distance $O(d(u, v))$, where the constant hidden in the big-O notation is determined by SINR parameters.

Lemma 4. *There exists a constant κ_0 (which depends merely of the SINR parameters) which satisfies the following property. Let u, v be a closest pair of nodes, $d(u, v) = d < 1/2$ in A. If u is the only transmitter in $\mathcal{B}(v, \kappa_0 \cdot d)$, then v receives the message from u.*

Using Lemma 4, the bound on the optimal size of witnessed strong selectors (Lemma 2) and the def. of a closest pair, one can obtaind the following corollaries.

Corollary 1. *There exists a constant κ (which depends merely of the SINR parameters) which satisfies the following property. Let u, v be a closest pair of nodes in A, $d(u, v) = d < 1/2$. Then, there exists a set $A' \subseteq A$ such that $u, v \in A'$, $|A'| \leq \kappa$ and v receives a message transmitted by u provided no other element of A' is sending a message at the same time.*

Corollary 2. *There exists a transmission schedule $\mathcal{S}$ of size $O(\log N)$ such that, for each closest pair (u, v) from a set A, u receives a message from v during an execution of $\mathcal{S}$ on A.*

[1] Note that there is no closest pair in A according to this definition if $d(x, y) > 1/2$ for each distinct $x, y \in A$.

4 High Level Idea of the Algorithm

Our token traversal algorithm builds a spanning tree of the communication graph of a network, where the source node s (the initial holder of the token) is the root. Each node, after receiving the token, transmits the broadcast and awake message. If a node u is awaken by v (i.e., u receives the first message from v), u becomes a *child* of v and v is the *parent* of u. After sending the broadcast and awake message, the token holder learns all its newly-awaken neighbors, who have become its children. After that, the token holder passes the token sequentially to all its children. Finally, it passes the token back to its parent. The algorithm ends when the source receives the token back from all its children. A similar approach, resembling both dfs and bfs, has appeared in the context of radio networks [6] under the name Breadth-Then-Depth (BTD) search.

The most challenging part for a design of the above strategy in the model considered in this paper is to learn the children of a token holder. To this aim, we consider the full selection problem: for a given node v and a set X of its neighbors unknown to v, the node v should learn the set X. Using appropriate novel selectors, and the idea of local leader election in the uniform SINR model [24], we can assure that full selection is done in $O(\log^2 N + |X| \log N)$ rounds. As each node becomes the child of only one other node, an application of full selection at each node (when it receives the token for the first time) would give $O\left(\sum_{v \in V}(\log^2 N + |\text{children}(v)| \log N)\right) = O(n \log^2 N)$ time algorithm. In order to improve time complexity to $O(n \log N)$, we will reduce full selection time from $O(\log^2 N + |X| \log N)$ to $O(\log N + |X| \log N)$. To this aim, we apply a kind of *implicit carrier sensing*. Thanks to that tool, we can check whether X is empty in $O(\log N)$ rounds and reduce complexity of full selection to $O((|X|+1) \log N)$.

Below, we discuss the technical ingredients of our solutions in more detail.

Network sparsification. As discussed above, each node v is supposed to determine its neighbors awaken by it after receiving the token, in the procedure called the full selection (Algorithm 5). To this aim we apply the sparsification technique presented in Sect. 5.2. Assume that $X \neq \emptyset$ is the set of neighbors of v which should be learnt by v. The idea is to execute a short schedule (of size $O(\log N)$) on X which guarantees that nodes from closest pairs (and possibly some other nodes) exchange messages. Then, a graph defined by successful exchanges of messages is built and a non-empty matching is determined in this graph. Finally, one node from each matched pair is chosen to be a member of a "sparsified" set. As such a procedure gives the sparsified set of size at most $|X|/2$, $r = O(\log |X|)$ repetitions of this procedure gives a set of size $O(1)$. Then, the only element(s) of the sparse set can report all r elements matched with them. Thus, r elements of X are reported in amortized time $O(\log N)$ per an element. In order to implement this technique efficiently under SINR constraints, we have introduced *witnessed strong selectors* (Sect. 3.1).

Implicit carrier sensing. As mentioned before, we use so-called implicit carrier sensing in order to quickly verify whether the set of nodes awaken by a given node v is (not) empty. More generally, implicit carrier sensing technique allows

for checking emptiness of a set $X \subseteq N_v$, provided two auxiliary nodes v_1, v_2 are known such that $d(v, v_1) \leq 1$, $d(v_1, v_2) \leq 1$ and $d(v, v_2) > 1$ (see Subsect. 5.1 for details). Our implementation of the token traversal algorithm will assure that $d(v, \text{parent}(v)) \leq 1$ and $d(v, \text{parent}(\text{parent}(v))) > 1$ for each v in graph-distance at least 2 from the source s. Thus, the auxiliary nodes can be $v_1 = \text{parent}(v)$ and $v_2 = \text{parent}(\text{parent}(v))$. This however does not apply to the source s (it does not have the parent) and its neighbors (there is no parent(parent(v)) for each $v \in N_s$). Therefore, we have to handle $\{s\} \cup N_s$ separately, using less efficient emptiness test and a more complex algorithm.

5 Implicit Carrier Sensing and Network Sparsification

In this section we introduce key tools applied in our token-traversal algorithm: implicit carrier sensing and sparsification. We present them separately, as we think they might be applicable in other problems in wireless ad hoc networks.

5.1 Implicit Carrier Sensing

Consider the problem that a node v is going to verify quickly whether some set $X \subseteq N_v$ is empty. Each node x knows whether $x \in X$ but nodes do not have any information regarding other elements of X. At the end of an execution of an algorithm, v should know whether $X = \emptyset$. This problem has been solved efficiently by so-called Echo procedure in the symmetric radio networks model, provided the node v knows some neighbor $w \notin X$ already [33]. We develop an analogous tool for SINR networks, which provides a limited carrier sense capability.

Assume that v, v_1, v_2 are fixed such that v is a neighbor of v_1 and v_1 is a neighbor of v_2. Moreover, at least one of distances $d(v, v_1)$, $d(v_1, v_2)$ is not smaller than $1/2$. Then, we can test emptiness of X by checking if

- v receives the message from v_1 when v_1 transmits together with X, and
- v_1 receives the message from v_2 when v_2 transmits together with X.

More precise description of the procedure is given as EmptinessTest below (see Algorithm 1). The first idea is that successful transmissions in both rounds correspond to the fact that X is empty. However, if X is small and v is very close to v_1 or v_1 is very close to v_2, it might be the case that interference from X does not prevent successful transmissions in both rounds under SINR constraints. Therefore, we need a more involved algorithm and analysis. The constant $c_{\alpha,\beta}$ in the algorithm is equal to the smallest number c such that c transmitting nodes located in distance (at most) 2 from a given node u produce interference which prevents reception by u of a message transmitted from distance $\geq 1/2$.

Algorithm 1. EmptinessTest(v, v_1, v_2, X)

Assumptions: $d(v, v_1) \leq 1$, $d(v_1, v_2) \leq 1$, $d(v, v_1) \geq 1/2$ or $d(v_1, v_2) \geq 1/2$, $X \subsetneq N_v$, $v_1, v_2 \notin X$.
Let $c_{\alpha,\beta}$ be the smallest natural number such that $\frac{P/(1/2)^\alpha}{\mathcal{N}+c_{\alpha,\beta}P/2^\alpha} < \beta$.

1: Round 1: v_1 and all elements of X transmit a message
2: Round 2: v_2 and all elements of X transmit a message
3: Round 3: if v_1 received a message in Round 2 then v_1 transmits a message
4: **if** v received a message in Round 1 **and** v received a message in Round 3 **then**
5: execute $(N, c_{\alpha,\beta})$-ssf on all elements of X
6: **if** v received a message: return false
7: **else** return true
8: **else**
9: return false

Lemma 5. *EmptinessTest works in $O(\log N)$ rounds. Moreover, if $d(v, v_1) \leq 1$ and $d(v_1, v_2) \leq 1$ and ($d(v, v_1) \geq 1/2$ or $d(v_1, v_2) \geq 1/2$) then EmptinessTest(v, v_1, v_2, X) returns true if and only if the set $X \subseteq N_v$ is empty.*

5.2 Network Sparsification

In this section we develop a tool for fast selection of elements of a set of nodes. The particular problem of *network sparsification* is as follows: given a non-empty set X of nodes inside $B(v, 1)$ such that at least two nodes are within distance $1/2$, choose a subset Y of X such that $1 \leq |Y| \leq |X|/2$. The idea is to use a short schedule which guarantees that close neighbors exchange messages (see Corollary 2), implicitly build a graph corresponding to these two-way transmissions, choose a non-empty matching in such a graph, and select one element from each matched pair.

As a direct application of Corollary 2 does not give a satisfying time complexity, we then introduce the notion of proximity graph and show how to build it with aid of witnessed strong selectors efficiently.

Exchange graphs. We define the notion of *exchange graph* which describes all possible exchange of messages between nodes during an execution of a schedule T. For a given schedule T and the set of nodes V, an *exchange graph* G_T is a graph on V, such that $\{u, w\}$ is an edge in G_T iff there is a successful transmission in both directions between u and w during T.

We say that a distributed protocol *builds* G_T if, as a result of an execution of this protocol on a given network, each node knows its neighbors in G_T. Note that, after a single execution of T, each node v knows nodes whose messages are successfully received by v. However, in order to determine its neighbors in G_T, v also needs to know which nodes received its message.

In order to provide this information to all nodes, we can apply the following algorithm, called ExGraphConstruction$_T$: first, each node v enumerates the senders $u_1, \ldots, u_p$ of all messages received during T; then, one can repeat $|T|$ times the schedule T, where each node transmits u_i in the ith repetition of T.

Lemma 6. *ExGraphConstruction$_T$ builds the exchange graph G_T in $O(|T|^2)$ rounds. Moreover, if the maximal degree δ of G_T is known to nodes in advance, the algorithm works in $O(|T|\delta)$.*

Proximity graphs. The idea behind our network sparsification algorithm is to build a graph on nodes of an input set X containing a closest pair as an edge, find a matching in that graph and choose one element of each matched pair as an element of the output Y. To do this, a fast protocol which produces a non-empty graph is needed, provided there is a closest pair in the input set of nodes. Let *proximity graph* of a given set of nodes be any graph on this set such that vertices of each closest pair u, v are connected by an edge (while the graph may contain more edges).

By Corollary 1 we know that, in an execution of (N, κ)-ssf, nodes of each closest pair exchange messages. Thus, by Lemma 6, ExGraphConstruction$_{\mathbf{T}}$ builds a proximity graph in $O(\log^2 N)$ rounds, where $\mathbf{T}$ is an (N, κ)-ssf of length $O(\kappa^2 \log N) = O(\log N)$ (see Theorem 1).

Our goal is to build a proximity graph faster. Our construction builds on the following observations. First, if u can hear v during an execution of T in a round in which w is transmitting as well, then u, w is for sure not a closest pair. Second, by Corollary 1, given a closest pair (u, v), u can hear v in a round in which v transmits and none of the other κ closest to u nodes transmits.

Given an (N, κ)-wss $\mathbf{S}$ for the constant κ from Corollary 1, one can build a proximity graph of degree $\kappa = O(1)$ in $O(\log N)$ rounds using the following distributed algorithm called ProximityGraphConstruction at a node v:

- Execute $\mathbf{S}$.
- Determine the set C_v of all nodes u such that v has received a message from u during $\mathbf{S}$ and v has not received any other message in rounds in which u is transmitting (according to $\mathbf{S}$).
- If $|C_v| > \kappa$, then remove all elements from C_v.
- Send information about the content of C_v to other nodes in consecutive $|C_v|$ repetitions of $\mathbf{S}$.
- Choose as neighbors in the final graph the set E_v of all elements $w \in C_v$ st $v \in C_w$.

Lemma 7. *Let $X \subseteq V$ be a set of nodes. Then ProximityGraphConstruction executed on X builds a proximity graph $H(X)$ of constant degree in $O(\log N)$ rounds.*

Handshakes and sparsification. Let $H(X)$ denote the proximity graph resulting from the ProximityGraphConstruction procedure executed by nodes from X. We assume that X contains nodes from a closest pair, thus $H(X)$ contains at least one edge (Lemma 7). Recall that our goal in this section is to choose a nonempty subset of X of size at most $|X|/2$. The idea is to build a non-empty matching on $H(X)$ and choose exactly one node per each matched pair. We say that nodes chosen by our procedure *survive*. For further applications, for each

node v which survives the procedure, we store its removed counterpart in the local variable $p(v)$.

Algorithm 2 finds a non-empty matching in a proximity graph $H(X)$ build by ProximityGraphConstruction (provided the set of edges of $H(X)$ is not empty) by connecting each pair of neighbors (v, w) such that v is the local minimum (its ID is smaller than IDs of its neighbors) and v has the smallest ID among neighbors of w in $H(X)$.

Algorithm 2. Handshake(X) ▷ Remark: an execution at $v \in X$

1: v executes ProximityGraphConstruction(v) using (N, κ)-wss **S** ▷ see L. 7
2: **if** $E_v = \emptyset$: v does not participate in further steps.
3: $\min_v \leftarrow \min_{u \in E_v}\{\mathrm{ID}(u)\}$
4: Execute **S**, where:
if $ID(v) < \min_v$: v transmits the message $m = \langle \underline{handshake}, ID(v), \min_v \rangle$
▷ v is a local minimum;
if $ID(v) > \min_v$: v transmits the message $m = \langle \underline{match}, ID(v), \min_v \rangle$
5: **if** $ID(v) < \min_v$ and v received the message $\langle \underline{match}, \min_v, ID(v), \rangle$ **then**
6: $p(v) \leftarrow \min_v$
7: status(v) $\leftarrow$ survived
8: **else**
9: status(v) $\leftarrow$ eliminated
10: v is switched off

Lemma 8. *Let $X \subseteq V$ be a subset of a network. Let $Y \subseteq X$ be the set of nodes that survived Handshake(X) (see line 7 of Algorithm 2). If there exists a closest pair in X then $1 \leq |Y| \leq |X|/2$. Moreover, for each $v \in Y$, $p(v) \in X \setminus Y$. The round complexity of Handshake procedure is $O(\log N)$.*

6 Token Traversal Algorithm

In this section we describe our token traversal algorithm. As it gives also immediate solution to the broadcasting problem, we present the algorithm in terms of the broadcasting task.

At the beginning of the main algorithm (Algorithm 3), the source s wakes up all its neighbors (which become its children). Then, the general idea is that each node v, after receiving the token, learns its children (nodes awaken by v) using FullSelection (Algorithm 5) and passes the token to them. As our time bound for FullSelection(v, X) for $v \in L_1$ and $X \subseteq N_v$ is $O(\log^2 N + |\mathrm{children}(v)| \log N)$ – see Lemma 10, this approach guarantees time $O(n \log^2 N)$ (and it might be $\Omega(n \log^2 N)$ if $|L_1| = \Omega(n)$). In order to achieve a better bound, the nodes from L_1 learn their children in a different way.

After the initial transmission by s, it learns the whole set of its neighbors $N_s = L_1$, using FullSelection. Then, s allows each $v \in N_s = L_1$ to transmit separately which wakes up all elements of L_2 and set the parent from L_1 for

each element of L_2. The goal of HandleSecondLayer is to select all elements of L_2, allow each of them to transmit separately which in turn gives information to each $w \in L_1$ about all its children. As mentioned earlier, we do not want to implement this task by calling FullSelection for each $v \in L_1$, as it would increase time complexity to the order of $n \log^2 N$. We postpone the exact description of HandleSecondLayer and discuss the remaining part of the algorithm and its subroutines. After handling the second layer, a standard token traversal algorithm starts from the source (Algorithm 4), where each node from $\{s\} \cup L_1$ already knows its children, while each other node $v \notin \{s\} \cup L_1$ learns children(v) using FullSelection.

Algorithm 3 contains pseudo-code of our main algorithm. Then, procedures called in the main algorithm are presented in the top-down fashion.

Remark. In pseudo-codes, we use set theoretic operations, e.g., $A \leftarrow X \setminus Y$. Such notation describes local decisions of nodes and means that each x knows whether it belongs to X and Y and therefore it can determine if it belongs to A.

Algorithm 3. BroadcastWithToken(s)

Initially for each node u parent(u) $= \perp$ and layer(v) $= 0$.
1: Transmit $\langle \underline{\text{hello}}, s \rangle$
2: FullSelection(s, L_1)
3: HandleSecondLayer
4: TokenTraversal(s)

if a node w receives $\langle \underline{\text{hello}}, v \rangle$ and parent(w) $= \perp$ **then**
    parent(w) $\leftarrow v$
    layer(w) $\leftarrow$ layer(v) $+ 1$

Algorithm 4. TokenTraversal(v)

1: Transmit $\langle \underline{\text{hello}}, v \rangle$
2: **if** layer(v) > 1 **then**
3: FullSelection($v, \{w \mid \text{parent}(w) = v\}$)
4: **for each** $w \in$ children(v) **do**
5: Transmit $\langle \underline{\text{token}}, w \rangle$ ▷ pass the token to w
6: Wait until receiving a message $\langle \underline{\text{release}}, v \rangle$ ▷ Token is back at v
7: Transmit $\langle \underline{\text{release}}, \text{parent}(v) \rangle$ ▷ pass the token to the parent of v

In the following, we describe the main subroutine FullSelection called at each node $v \notin L_1$. FullSelection(v, X) repeats procedure PartialSelection several times, until all elements of X are selected, i.e., each of them transmits a message received by v. An execution of PartialSelection(v, X) results in reporting $r > 0$

elements of X in $O(r \log N)$ rounds, provided X is not empty. In the case that X is empty, PartialSelection(v, X) ends in $O(\log N)$ rounds when $v \notin \{s\} \cup L_1$ and in $O((\log n) \cdot (\log N))$ rounds otherwise.

The procedure PartialSelection (Algorithm 6) executes Handshake several times. (An alternative for partial selection might be e.g. by $(N, k, k/2)$-selectors [4] for $k = 2, 4, 8, \ldots, 2^{\log n}$. However, no constructive solutions with optimal size are known for them.) Handshake(X) sparsifies the set X. As Handshake is always executed on $X \subseteq N_v$ for a reference node v, X is contained in a ball of radius 1. Let $m = |X|$. If m is smaller than $\chi(1/2, 2)$, then each element of X transmits separately (see line 6 of Algorithm 6). Otherwise, there exists a closest pair of nodes in X, and some elements are removed from X such that the size of X after the execution is in the range $[1, m/2]$ (see Lemma 8). In this way at least one element of X is selected in $O(\log |X|)$ executions of Handshake.

However, our goal is to select one element per each execution of Handshake on the average. Fortunately, each node v which survives the ith execution of Handshake has associated the unique element $p(v)$ which has survived the first $i-1$ executions of Handshake and has not survived the ith execution (see Lemma 8). The node v stores such elements in $P(v)$.

Algorithm 5. FullSelection(v, X) ▷ v learns all elements of X

1: $Y \leftarrow X$, $w \leftarrow v$, children$(v) \leftarrow \emptyset$
2: **while** $w \neq \perp$ **do**
3: $w \leftarrow$ PartialSelection(v, X)
4: $Y \leftarrow P(w)$ ▷ if $w \neq \perp$, w broadcasts $P(w)$ in $|P(w)| + 1$ rounds
5: $X \leftarrow X \setminus Y$
6: children$(v) \leftarrow$ children$(v) \cup Y$ ▷ v learns Y in step 4

Lemma 9. *1. Assume that $X \subseteq N_v$ is not empty. Then, PartialSelection(v, X) is finished after $O(r \log N)$ rounds for $0 < r \leq \log n$ and v has received a message from $w \in X$ such that $P(w) \subseteq X$ and $|P(w)| = r$.*
2. PartialSelection(v, X) for $X = \emptyset$ works in $O(\log N)$ rounds for $v \notin L_0 \cup L_1$ and in $O((\log n) \cdot (\log N))$ rounds for $v \in L_0 \cup L_1$. Moreover, v is aware of the fact that $X = \emptyset$ after the execution of PartialSelection(v, X) for $X = \emptyset$.

To summarize, we state the following properties of FullSelection.

Lemma 10. *Let $X \subseteq N_v$. Then, v knows all elements of X after an execution of FullSelection(v, X). Moreover, each $u \in X$ transmits uniquely at some round of FullSelection(v, X). The execution time is $O(|X| \log N + f(N))$, where $f(N) = O(\log N)$ if $v \notin L_0 \cup L_1$ and $f(N) = O((\log n) \cdot (\log N))$ otherwise.*

Handling the second layer. Recall that each node from $L_0 \cup L_1$ is the only transmitter in some round during steps 1. and 2. of the main algorithm (Algorithm 3). The nodes from L_2 are awaken in this way and they know their parents from L_1.

Algorithm 6. PartialSelection(v, X) ▷ Assumption: $X \subseteq N_v$

1: **if** $v \notin L_0 \cup L_1$ **then** ▷ $L_0 = \{s\}$, $L_1 = N_s$
2: **if** EmptinessTest$(v, \text{parent}(v), \text{parent}(\text{parent}(v)), X)$: return $\perp$
3: $r \leftarrow 1$
4: **for each** $w \in X$**:** $P(w) \leftarrow \emptyset$
5: **repeat**
6: Execute (N, k)-ssf on X for $k = \lceil \chi(1/2, 1) \rceil + 1$.
7: **if** v received a message from some node w during step 6 **then**
8: return w ▷ i.e., v transmits a message which ends the procedure
9: Handshake(X)
10: **for each** w**: if** w survived Handshake(X): $P(w) \leftarrow P(w) \cup \{p(w)\}$
11: **for each** w**: if** w did not survive Handshake(X): w remove itself from X
12: $r \leftarrow r + 1$
13: **until** $r = \log N$ ▷ until $r = \log n$ if n is known
14: return $\perp$

Now, we describe HandleSecondLayer subroutine which assures that each node $v \in L_2$ is a unique transmitter in some round (and it transmits ID of its parent in each transmission). In this way the nodes from L_1 learn about their children.

To achieve the above stated goal, we repeat the following procedure. First, the leader v in L_2 is elected and, all elements of $N_v \cap L_2$ are selected using FullSelection$(v, N_v \cap L_2)$. Then, all selected elements are removed from consideration and the process is repeated until no unselected elements in L_2 remain. As the consecutive leaders are in distance > 1 to each other, this process finishes after at most $\chi(1, 2)$ elections of the leader. More formal presentation of this idea is given in Algorithm 7.

Algorithm 7. HandleSecondLayer ▷ Assumption: each v knows if $v \in L_2$

1: $L \leftarrow L_2$ ▷ Initially, L is the second layer
2: $c \leftarrow \chi(1, 2)$
3: **for** i=1,2,. . . ,c **do**
4: $v \leftarrow$ Leader(L)
5: v transmits $\langle \underline{\text{leader}}, v \rangle$
6: $X \leftarrow \{w \mid w$ received the message $\underline{\text{leader}}\}$
7: FullSelection(v, X)
8: $L \leftarrow L \setminus X$

Now, we provide an efficient implementation of leader election in line 4 of Algorithm 7. We require that exactly one element $x \in X$ has the status *leader* as a result of a *leader election* algorithm. (Observe that we do not require that all elements of X know ID of the node with status *leader*.)

The idea of the leader election procedure (Algorithm 8) is to repeat Handshake several times in order to sparsify the input set X, as in PartialSelection. In this way, some node $w \in X$ will be the only transmitter at some round, after $O(\log N)$ repetitions of Handshake. The problem is that the unique transmitter w might be unaware of its uniqueness. If all elements of X were also in N_v for a distinguished node v, then v could confirm reception of a message from w and inform in this way w that it was the leader[2]. As the set of nodes $X \subset L_2$ executing the leader election procedure is not necessarily a subset of N_v for any v, PartialSelection does not give a solution to the leader election problem directly. Instead, we use the fact that each node $v \in X$ knows parent$(v) \in L_1$. We assure that each node $v \in L_1$ that hears its child $w \in L_2$, tries to report this fact to the source. If the source node receives such a message at some time, it chooses w to be the leader and passes this information through $v = \text{parent}(w)$ to w. This is guaranteed to happen when w was the unique transmitter in L_2, and it will happen eventually for some $w \in L_2$ in at most $\log n$) executions of the for-loop in Algorithm 8.

Algorithm 8. Leader(L) ▷ Assumption: parent$(x) \in L_1$, $x \in L_2$ for each $x \in L$

1: leader$(v) \leftarrow$ false for all $v \in L$
2: elected $\leftarrow$ false ▷ elected is a local variable stored at s
3: **for** $i = 1, 2, \ldots, \log n$ **do**
4: Execute (N, k)-ssf for $k = \lceil \chi(1/2, 2) \rceil + 1$, each transmission is followed by:
Round 1:
if $w \in L_1$ received a message from its child $x \in L_2$:
w transmits $\langle \underline{\text{leader-proposal}}, x, w \rangle$
Round 2:
if elected=false and s received $\langle \underline{\text{leader-proposal}}, x, y \rangle$:
s transmits $\langle \underline{\text{leader-elect}}, x, y \rangle$; elected $\leftarrow$ true
Round 3:
if $w \in L_1$ received a message $\langle \underline{\text{leader-elect}}, x, w \rangle$: w transmits $\langle \underline{\text{leader-elect}}, x \rangle$.
5: **if** $x \in L$ received a message $\langle \underline{\text{leader-elect}}, x \rangle$ in Round 3: leader$(x) \leftarrow$ true
6: Handshake(L)
7: $L \leftarrow$ nodes from L which survived Handshake(L)

Proposition 1. *Let $L \subseteq L_2$ be a set of nodes such that parent(parent(x)) $= s$ and parent$(x) \in L_1 \cap N_x$ for each $x \in L$. Then, Leader(L) solves the leader election problem on L in $O((\log N) \cdot (\log n))$ rounds.*

Given the leader election procedure, we can prove that each node from L_2 is the only transmitter in some round of HandleSecondLayer. This in turn gives information to nodes from L_1 about their children.

[2] Note that, under SINR constraints, v can receive a message from some node x even when x is not the unique transmitter in a round. However, as v "selects" the leader and announces its choice, this does not cause any problem with uniqueness of the leader.

Lemma 11. *Assume that parent$(v) = s$ for each $v \in L_1$ and parent$(u) \in L_1 \cap N_u$ for each $u \in L_2$. Then, each $u \in L_2$ is the only transmitter in some round of an execution of HandleSecondLayer. Moreover, HandleSecondLayer works in in $O(n \log N)$ rounds. otherwise.*

Theorem 1. *BroadcastWithToken solves weak connectivity broadcast in the ad hoc SINR model in $O(n \log N)$ rounds.*

Proof. Lemmas 10 and 11 imply that steps 1–3 of the algorithm are finished in time stated in the theorem. Then, the TokenTraversal algorithm builds a spanning tree of a network, the token is passed once over each edge of this tree in each direction which altogether takes $O(n)$ rounds. Moreover, for each node $v \notin L_0 \cup L_1$, FullSelection(v, X) is executed when v receives the token for the first time, for X equal to the set of children of v. An execution of FullSelection(v, X) takes $O((|X|+1) \log N)$ rounds, where X is the set of selected elements. As each $w \in V \setminus (L_0 \cup L_1)$ is only once an element of X in an execution of FullSelection(v, X) (when $v =$ parent(w)), the overall complexity of all executions of FullSelection during TokenTraversal(s) is $O(n \log N)$.

7 Lower Bound and Extensions

Lower bound. Employing a similar approach as in [5], we build a network of linear diameter such that it takes at least $\Omega(n \log N)$ rounds to broadcast a message.

Theorem 2. *For any deterministic algorithm $\mathcal{A}$ and $n < N/6$, there exists a network $\mathbf{N}$ of size $3n+1$ on the plane such that it takes $\Omega(n \log N)$ rounds before $\mathcal{A}$ completes the broadcast in $\mathbf{N}$ in a weakly connected SINR network.*

Randomized algorithm. In [10] the authors proposed a randomized algorithm that solves broadcast in time $O(n \log^2 n)$ and a lower bound of $\Omega(n)$. Our result fits into the scenario presented therein provided each node picks a random ID in range $[1, n^3]$ and performs the deterministic algorithm, which works as long as the IDs are different (this is true with high probability). Thus, our solution is $O(n \log n)$.

Constructive solution. We need (N, k)-wss only for constant values of parameter k. Thus, by Lemma 3, the actual algorithm for fixed N can be determined in time polynomial with respect to N.

8 Conclusions

We presented a novel token traversal algorihm in weakly-connected ad hoc networks under the uniform-power SINR model, leading to asymptotically optimal deterministic spanning tree and broadcast algorithm and a nearly-optimal randomized solution improving [10]. The token traversal occurred to be efficient for spanning tree and broadcast problem, therefore we conjecture that it could play a substantial role in algorithmics in general class of ad hoc wireless networks.

Acknowledgments. The authors would like to thank Darek Kowalski for fruitful discussions and his comments on the paper.

References

1. Alon, N., Bar-Noy, A., Linial, N., Peleg, D.: A lower bound for radio broadcast. J. Comput. Syst. Sci. **43**(2), 290–298 (1991)
2. Bar-Yehuda, R., Goldreich, O., Itai, A.: On the time-complexity of broadcast in multi-hop radio networks: an exponential gap between determinism and randomization. J. Comput. Syst. Sci. **45**(1), 104–126 (1992)
3. Bienkowski, M., Jurdzinski, T., Korzeniowski, M., Kowalski, D.R.: Distributed online and stochastic queuing on a multiple access channel. In: Aguilera, M.K. (ed.) DISC 2012. LNCS, vol. 7611, pp. 121–135. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33651-5_9
4. Bonis, A.D., Gasieniec, L., Vaccaro, U.: Optimal two-stage algorithms for group testing problems. SIAM J. Comput. **34**(5), 1253–1270 (2005)
5. Bruschi, D., Pinto, M.D.: Lower bounds for the broadcast problem in mobile radio networks. Distrib. Comput. **10**(3), 129–135 (1997)
6. Chlebus, B.S., Kowalski, D.R., Pelc, A., Rokicki, M.A.: Efficient distributed communication in ad-hoc radio networks. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011. LNCS, vol. 6756, pp. 613–624. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22012-8_49
7. Clementi, A.E.F., Crescenzi, P., Monti, A., Penna, P., Silvestri, R.: On computing ad-hoc selective families. In: Goemans, M., Jansen, K., Rolim, J.D.P., Trevisan, L. (eds.) APPROX/RANDOM-2001. LNCS, vol. 2129, pp. 211–222. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44666-4_24
8. Clementi, A.E.F., Monti, A., Silvestri, R.: Selective families, superimposed codes, and broadcasting on unknown radio networks. In: SODA 2001, pp. 709–718 (2001)
9. Czumaj, A., Rytter, W.: Broadcasting algorithms in radio networks with unknown topology. In: FOCS, pp. 492–501. IEEE Computer Society (2003)
10. Daum, S., Gilbert, S., Kuhn, F., Newport, C.: Broadcast in the ad hoc SINR model. In: Afek, Y. (ed.) DISC 2013. LNCS, vol. 8205, pp. 358–372. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41527-2_25
11. De Marco, G.: Distributed broadcast in unknown radio networks. SIAM J. Comput. **39**(6), 2162–2175 (2010)
12. Emek, Y., Gasieniec, L., Kantor, E., Pelc, A., Peleg, D., Su, C.: Broadcasting in UDG radio networks with unknown topology. Distr. Comput. **21**(5), 331–351 (2009)
13. Goussevskaia, O., Moscibroda, T., Wattenhofer, R.: Local broadcasting in the physical interference model. In: DIALM-POMC 2008, pp. 35–44. ACM (2008)
14. Halldórsson, M.M., Holzer, S., Lynch, N.A.: A local broadcast layer for the SINR network model. In: PODC 2015, ADM, pp. 129–138 (2015)
15. Halldórsson, M.M., Mitra, P.: Nearly optimal bounds for distributed wireless scheduling in the SINR model. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011. LNCS, vol. 6756, pp. 625–636. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22012-8_50
16. Halldórsson, M.M., Mitra, P.: Towards tight bounds for local broadcasting. In: Kuhn, F., Newport, C.C. (eds.) FOMC, p. 2. ACM (2012)
17. Halldórsson, M.M., Tonoyan, T.: How well can graphs represent wireless interference? In: STOC 2015, pp. 635–644 (ACM)

18. Hobbs, N., Wang, Y., Hua, Q.-S., Yu, D., Lau, F.C.M.: Deterministic distributed data aggregation under the SINR model. In: Agrawal, M., Cooper, S.B., Li, A. (eds.) TAMC 2012. LNCS, vol. 7287, pp. 385–399. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29952-0_38
19. Jurdzinski, T., Kowalski, D.R.: Distributed backbone structure for algorithms in the SINR model of wireless networks. In: Aguilera, M.K. (ed.) DISC 2012. LNCS, vol. 7611, pp. 106–120. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33651-5_8
20. Jurdzinski, T., Kowalski, D.R.: Distributed randomized broadcasting in wireless networks under the SINR model. In: Kao, M.-Y. (ed.) Encyclopedia of Algorithms, pp. 577–580. Springer, New York (2016). https://doi.org/10.1007/978-1-4939-2864-4_604
21. Jurdzinski, T., Kowalski, D.R., Rozanski, M., Stachowiak, G.: Distributed randomized broadcasting in wireless networks under the SINR model. In: Afek, Y. (ed.) DISC 2013. LNCS, vol. 8205, pp. 373–387. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41527-2_26
22. Jurdzinski, T., Kowalski, D.R., Rozanski, M., Stachowiak, G.: On the impact of geometry on ad hoc communication in wireless networks. In: PODC 2014, pp. 357–366. ACM (2014)
23. Jurdzinski, T., Kowalski, D.R., Stachowiak, G.: Distributed deterministic broadcasting in uniform-power ad hoc wireless networks. In: Gąsieniec, L., Wolter, F. (eds.) FCT 2013. LNCS, vol. 8070, pp. 195–209. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40164-0_20
24. Jurdzinski, T., Kowalski, D.R., Stachowiak, G.: Distributed deterministic broadcasting in wireless networks of weak devices. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M., Peleg, D. (eds.) ICALP 2013. LNCS, vol. 7966, pp. 632–644. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39212-2_55
25. Kantor, E., Lotker, Z., Parter, M., Peleg, D.: The minimum principle of SINR: a useful discretization tool for wireless communication. In: FOCS 2015, pp. 330–349. IEEE (2015)
26. Kantor, E., Lotker, Z., Parter, M., Peleg, D.: The topology of wireless communication. J. ACM **62**(5), 37:1–37:32 (2015)
27. Kesselheim, T.: A constant-factor approximation for wireless capacity maximization with power control in the SINR model. In: SODA 2011, pp. 1549–1559. SIAM (2011)
28. Kesselheim, T.: Dynamic packet scheduling in wireless networks. In: PODC 2012, pp. 281–290. ACM (2012)
29. Kesselheim, T., Vöcking, B.: Distributed contention resolution in wireless networks. In: Lynch, N.A., Shvartsman, A.A. (eds.) DISC 2010. LNCS, vol. 6343, pp. 163–178. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15763-9_16
30. Kowalski, D.R.: On selection problem in radio networks. In: PODC 2005, pp. 158–166. ACM (2005)
31. Kowalski, D.R., Moses Jr., W.K., Vaya, S.: Deterministic backbone creation in an SINR network without knowledge of location. CoRR, abs/1702.02460 (2017)
32. Kowalski, D.R., Pelc, A.: Broadcasting in undirected ad hoc radio networks. In: PODC 2003, pp. 73–82. ACM (2003)
33. Kowalski, D.R., Pelc, A.: Faster deterministic broadcasting in ad hoc radio networks. SIAM J. Discrete Math. **18**(2), 332–346 (2004)
34. Kowalski, D.R., Pelc, A.: Time of deterministic broadcasting in radio networks with local knowledge. SIAM J. Comput. **33**(4), 870–891 (2004)

35. Kushilevitz, E., Mansour, Y.: An $\Omega(DLog(N/D))$ lower bound for broadcast in radio networks. In: PODC 1993, pp. 65–74. ACM (1993)
36. Moscibroda, T., Wattenhofer, R.: The complexity of connectivity in wireless networks. In: INFOCOM 2006. IEEE (2006)
37. Yu, D., Hua, Q., Wang, Y., Lau, F.C.M.: An o(log n) distributed approximation algorithm for local broadcasting in unstructured wireless networks. In: DCOSS 2012, pp. 132–139. IEEE (2012)