Token Traversal in Ad Hoc Wireless Networks via Implicit Carrier Sensing*

Tomasz Jurdziński[1], Michał Różański[1], and Grzegorz Stachowiak[1]

[1]Institute of Computer Science, University of Wrocław, Poland.
E-mail:tju@cs.uni.wroc.pl (corresponding author)

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*. In such networks, connectivity is defined based on highly reliable links, that is, where both ends are located far closer from their transmission boundaries. What happens if the network is not strongly-connected, e.g., it contains some long but still viable "shortcut links" connecting transmission boundaries? It is known that 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 result holds even if the network is of constant diameter, algorithms may use randomization, nodes of the network have access to their own coordinates and have carrier sensing capabilities. The best up-to-date (randomized) distributed algorithm, designed by Daum et al. [12], accomplishes broadcast task in $O(n \log^2 n)$ communication rounds with high probability.

In this work, inspired by the work on broadcasting, we show a novel deterministic distributed implementation of token traversal — a fundamental tool in distributed systems — 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 some integer $N \geq n$. This result implies an $O(n \log n)$-round randomized solution that does not require IDs, which improves the result from [12]. 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, efficient in terms of time and memory, is based on a novel implicit algorithmic carrier sensing method and a new type of selectors, which might be of independent interest and applicable to other communication tasks in distributed ad hoc setting.

Keywords: Wireless networks, SINR, Ad hoc networks, Token traversal, Broadcast, Lower bounds, Deterministic and randomized algorithms, Algorithmic carrier sensing, Selectors, BTD trees

*This work was supported by the Polish National Science Centre grants DEC-2012/06/M/ST6/00459, DEC-2012/07/B/ST6/01534, and 2014/13/N/ST6/01850.

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 capability and knowledge of stations 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. Token traversal has also been studied in the context of broadcasting and other similar communication tasks. 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 rare 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 tree, along participated nodes, 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. Presented algorithm builds a spanning tree of a network and uses a specially implemented general token traversal technique. 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 Section 7), which improves the $O(n \log^2 n)$ algorithm of Daum et al. [12].

Our results show that even a single broadcast benefits from the token traversal technique, and there is no other technique that could accomplish broadcast asymptotically faster in weakly-connected networks. This suggests that techniques based on token traversal could play important role in efficient algorithmics on weakly-connected ad hoc networks. 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 [34], 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 [2, 18, 26, 27] and algorithm design [37, 14, 4, 12, 15, 20, 22, 25, 38, 28, 16, 19]. First wave of algorithmic research on communication under SINR constraints focused on local problems problems. This includes in particular the local broadcast and link scheduling [14, 17, 4, 38, 29, 30].

Efficient implementation of tokens were provided in more restricted models of wireless ad hoc networks. Token-based algorithms were considered in related models of multiple-access channel and radio networks, e.g., [5, 32]. In radio networks, an $O(\log N)$ procedure of token passing was presented in [35], relying on Echo procedure. In [8] it was combined with the BTD tree traversal to solve various communication tasks. In the SINR model of weak devices (i.e., when devices can receive messages only from nodes located within distance substantially smaller than transmission range), subsumed by and less complex then the weak-connectivity model, efficient implementation of a token was provided [32].

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 [25, 24], where D is the diameter of the communication graph. The randomized results on broadcast in ad hoc settings include papers of Daum et al. [12] and Jurdziski et al. [23]. 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. [15] proposed an algorithm which can be faster assuming that stations are equipped with some extra capabilities, including carrier sensing.

For weak connectivity networks Daum et al. have provided an example showing that any distributed (randomized) algorithm needs $\Omega(n)$ time in order to finish broadcast, even in 2-broadcastable networks (i.e., networks for which two rounds are sufficient to finish the broadcast task). 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, 3, 11, 36, 31, 33, 13].

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 stations transmitting at the same time. *Transmission range* is the maximal distance at which a station 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* [12, 21].

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 stations Each node has a unique identifier from the set $[N]$, where $N > n$ and n is the number of nodes in the network. We consider the case that the value of n (or its linear approximation) is known to nodes of a network as well as the scenario, where no information about the value of n is available. 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 Section 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. Thus, for given parameters (including the size of the network n), we are interested in the largest time among networks with these parameters.

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, even though some procedures, e.g., new witnessed strong selectors, could be non-constructive in general — this is because of the fact that our algorithms use such procedures 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 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 ID(v) in $[N]$ and its coordinates on the plane. In descriptions of algorithms, 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. *[10] 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 $y \notin X$ 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. [9], 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. This fact has been used in various papers several times, we recall it here as we have not found an explicit statement which we need in other papers.

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.*

The following corollaries following from Lemma 4, the optimal size of witnessed strong selectors (Lemma 2) and the definition of a closest pair.

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 from u provided u is sending a message and no other element of A' is sending a message.*

Corollary 2. *There exists a transmission schedule $\mathcal{S}$ of size $O(\log N)$ such that, for each closest pair (u, v) in A, u receives a message from v during an execution of $\mathcal{S}$ on the set 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 [8] 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 node. 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 [25], 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* (see Subsection 5.1). Thanks to that tool, we can check whether X is empty in $O(\log N)$ rounds and reduce complexity of full selection to $O((|X| + 1) \log N)$.

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 $d(v, v_2) > 1$. Our implementation

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

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 $\text{parent}(\text{parent}(v))$ for each $v \in N_s$). Therefore, we have to handle $\{s\} \cup N_s$ separately, using less efficient emptiness test and more complex algorithm.

5 Implicit Carrier Sensing and Network Sparsification

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. This gave a kind of implicit collision detection and lead to surprisingly efficient algorithms in radio networks without collision detection [34]. 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 (see Fig. 1)

- 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 Alg. 1). 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 $\frac{P/(1/2)^\alpha}{\mathcal{N} + c_{\alpha,\beta} P/2^\alpha} < \beta$.
Round 1: $v_1$ and all elements of $X$ transmit a message
Round 2: $v_2$ and all elements of $X$ transmit a message
Round 3: if $v_1$ received a message in Round 2 then $v_1$ transmits a message
if $v$ received a message in Round 1 and $v$ received a message in Round 3 then
    execute $(N, c_{\alpha,\beta})$-ssf on all elements of $X$
    if $v$ received a message: return false
    else return true
else
    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 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 can hear each other (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. This, in turn, gives a $O(\log N)$ time algorithm for network sparsification.

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 stations 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 station knows its neighbors in G_T. Note that, after a single execution of T, each station v knows stations whose messages are successfully received by v in T. However, in order to determine its neighbors in G_T, v also needs to know which nodes received its message during an execution of T. In order to provide this information to all stations, we can apply the following algorithm, called ExGraphConstruction$_T$. First, each station 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 station 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 stations in advance, the algorithm works in $O(|T|\delta)$.*

Proximity graphs The idea behind our network sparsification algorithm it to build a non-empty graph on nodes of an input set X containing a closest pair, 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 hear each other. Thus, by Lemma 6 the protocol ExGraphConstruction$_{\mathbf{T}}$, where $\mathbf{T}$ is an (N, κ)-ssf of length $O(\kappa^2 \log N)$ (see Theorem 1) builds a proximity graph in $O(\log^2 N)$ rounds.

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 stations transmits.

Given an (N, κ)-wss $\mathbf{S}$ for the constant κ from Corollary 1, one can build a proximity graph in $O(\log N)$ rounds using the following distributed algorithm called ProximityGraphConstruction at a station v (see pseudocode in Alg. 9 and an illustration on Fig. 2):

- Execute $\mathbf{S}$.
- Determine the set C_v of all stations 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 in X. We assume that X contains a closest pair, thus $H(X)$ contains at least one edge (Lemma 7). Our goal in this section is to choose a nonempty subset of X of size at most $|X|/2$ The general 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 matching in a proximity graph $H(X)$ build by Alg. 9 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 $H(X)$ (see an example on Fig. 3).

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

```
1: Each $v \in X$ executes ProximityGraphConstruction($v$) using $(N, \kappa)$-wss $\mathbf{S}$.   ▷ see Lemma 7
2: **if** $E_v = \emptyset$: $v$ does not participate in further steps.
3: $\min_v \leftarrow \min_{u \in E_v}\{ID(u)\}$
4: Execute $\mathbf{S}$, 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 broadcasting task.

At the beginning of the main algorithm (Alg. 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 (Alg. 5). As our time bound for FullSelection(v, X) for $v \in L_1$ is $O(\log^2 N + |\text{children}(v)| \log N)$, 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 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 of them. 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 (Alg. 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 informal set theoretic operations, e.g., $A \leftarrow X \setminus Y$. Such notation describes local decisions of nodes and means that each node 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($v$) $= 0$.
Transmit $\langle \underline{\text{hello}}, s \rangle$
FullSelection($s, L_1$)
HandleSecondLayer
TokenTraversal($s$)

**if** a station $w$ 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)

```
Transmit $\langle \underline{\text{hello}}, v\rangle$
if layer($v$) $> 1$ then
    FullSelection($v, \{w \mid \text{parent}(w) = v\}$)
for each $w \in$ children($v$) do
    Transmit $\langle \underline{\text{token}}, w\rangle$.                      ▷ pass the token to $w$
    Wait until receiving a message $\langle \underline{\text{release}}, v\rangle$.   ▷ Token is back at $v$
Transmit $\langle \underline{\text{release}}, \text{parent}(w)\rangle$.          ▷ 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 $O(\log^2 N)$ rounds otherwise.

The procedure PartialSelection (Alg. 6) executes Handshake several times. (An alternative for partial selection might be e.g. by $(N, k, k/2)$-selectors [6] for $k = 2, 4, 8, \ldots, 2^{\log n}$. However, no constructive solutions with optimal size are known for them.) Handshake(X) is 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 information about $P(w)$ 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
```

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($v$, parent($v$), parent(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 execution of PartialSelection
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 $\perp$
```

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 $O(\log^2 N)$ rounds for $v \in L_0 \cup L_1$ ($O(\log n) \cdot (\log N)$) when n is known). Moreover, v is aware of the fact that $X = \emptyset$ after the execution of PartialSelection(v, X) for $X = \emptyset$.

To summarize the above observations 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 $f(N) = O(\log^2 N)$ otherwise. (If n is known, then $f(N) = O(\log^2 N)$ is replaced with $f(N, n) = O((\log N) \cdot (\log n))$.)*

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 (Alg. 3). The nodes from L_2 are awaken in this way and they know their parents (note that parent$(v) \in L_1$ for each $v \in L_2$).

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 described 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 should finish after at most $\chi(1, 2)$ elections of the leader. More formal presentation of this idea is given in Alg. 7.

Algorithm 7 HandleSecondLayer

```
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 Alg. 7. First, define the problem that needs to be solved. The *leader election* problem is solved for a given non-empty set of nodes X if exactly one element $x \in X$ has the status *leader*. (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 (Alg. 8) is to repeat Handshake several times in order to sparsify the input set X, as in PartialSelection. This way, some node $w \in X$ will be the only transmitter at some round. The problem is that the unique transmitter w might be unaware of its uniqueness. If all elements of X were included in N_v for a distinguished node v, then v would confirm reception of a message from w and inform in this way w (and all elements of X) about the ID of the leader.[2] Therefore, we cannot apply PartialSelection directly. Instead, we use the fact that $X \subseteq L_2$ and each node $v \in X$ knows parent$(v) \in L_1$. We assure that each node $v \in L_1$ that hears its child from $w \in L_2$ tries to report it to the source. If the source node receives such message at some time, it chooses w to be the leader and passes this information through $v =$ 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 $w \in L_2$ (see Fig. 4(a)-(b)).

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

```
Algorithm 8 Leader($L$)                ▷ Assumption: parent($x$) $\in L_1$, $x \in L_2$ for each $x \in L$
leader($v$) $\leftarrow$ false for all $v \in L$
elected $\leftarrow$ false                                   ▷ elected is a local variable stored at $s$
for $i = 1, 2, \ldots, \log N$ do                           ▷ $i = 1, 2, \ldots, \log n$ when $n$ is known
    Execute $(N, k)$-ssf for $k = \lceil \chi(1/2, 2) \rceil + 1$, where each transmission is followed by:
    Round 1:
    if $w \in L_1$ received a message from its child $x \in L_2$: $w$ transmits $\langle$leader-proposal$, x, w\rangle$
    Round 2:
    if elected=false and $s$ received $\langle$leader-proposal$, x, y\rangle$:
        $s$ transmits $\langle$leader-elect$, x, y\rangle$; elected $\leftarrow$ true
    Round 3:
    if $w \in L_1$ received a message $\langle$leader-elect$, x, w\rangle$: $w$ transmits $\langle$leader-elect$, x\rangle$.
    if $x \in L$ received a message $\langle$leader-elect$, x\rangle$ in Round 3: leader($x$) $\leftarrow$ true
    Handshake($L$)
    $L \leftarrow$ 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 $O(\log^2 N)$ rounds if n is unknown and in $O((\log N)(\log n))$ rounds otherwise.*

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.

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 $O((n + \log N) \log N)$ rounds if n is unknown and in $O(n \log N)$ rounds otherwise.*

Theorem 1. *The BroadcastWithToken algorithm solves weak connectivity broadcast in the ad hoc SINR model in $O(n \log N)$ rounds if n is known and in $O(n \log N + \log^2 N)$ otherwise.*

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 [7] we construct 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 $3n + 1$ such that it takes $\Omega(n \log N)$ rounds before $\mathcal{A}$ completes the broadcast in $\mathbf{N}$.*

Randomized Algorithm. In [12] 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.

Other extensions. Extensions regarding optimization of local memory and acknowledged vs. non-acknowledged broadcast are deferred to the full version, see Appendix.

References

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

[2] B. Aronov and M. J. Katz. Batched point location in SINR diagrams via algebraic tools. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 65–77, 2015.

[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. Syst. Sci.*, 45(1):104–126, 1992.

[4] L. Barenboim and D. Peleg. Nearly optimal local broadcasting in the SINR model with feedback. In *Structural Information and Communication Complexity - 22nd International Colloquium, SIROCCO 2015, Montserrat, Spain, July 14-16, 2015, Post-Proceedings*, pages 164–178, 2015.

[5] M. Bienkowski, T. Jurdzinski, M. Korzeniowski, and D. R. Kowalski. Distributed online and stochastic queuing on a multiple access channel. In *Distributed Computing - 26th International Symposium, DISC 2012, Salvador, Brazil, October 16-18, 2012. Proceedings*, pages 121–135, 2012.

[6] A. D. Bonis, L. Gasieniec, and U. Vaccaro. Optimal two-stage algorithms for group testing problems. *SIAM J. Comput.*, 34(5):1253–1270, 2005.

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

[8] B. S. Chlebus, D. R. Kowalski, A. Pelc, and M. A. Rokicki. Efficient distributed communication in ad-hoc radio networks. In *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part II*, pages 613–624, 2011.

[9] A. E. F. Clementi, P. Crescenzi, A. Monti, P. Penna, and R. Silvestri. On computing ad-hoc selective families. In *RANDOM-APPROX 2001, Berkeley, CA, USA, August 18-20, 2001, Proceedings*, pages 211–222, 2001.

[10] A. E. F. Clementi, A. Monti, and R. Silvestri. Selective families, superimposed codes, and broadcasting on unknown radio networks. In S. R. Kosaraju, editor, *SODA*, pages 709–718. ACM/SIAM, 2001.

[11] A. Czumaj and W. Rytter. Broadcasting algorithms in radio networks with unknown topology. In *FOCS*, pages 492–501. IEEE Computer Society, 2003.

[12] S. Daum, S. Gilbert, F. Kuhn, and C. C. Newport. Broadcast in the ad hoc SINR model. In Y. Afek, editor, *Distributed Computing - 27th International Symposium, DISC 2013, Jerusalem, Israel, October 14-18, 2013. Proceedings*, volume 8205 of *Lecture Notes in Computer Science*, pages 358–372. Springer, 2013.

[13] Y. Emek, L. Gasieniec, E. Kantor, A. Pelc, D. Peleg, and C. Su. Broadcasting in udg radio networks with unknown topology. *Distributed Computing*, 21(5):331–351, 2009.

[14] O. Goussevskaia, T. Moscibroda, and R. Wattenhofer. Local broadcasting in the physical interference model. In M. Segal and A. Kesselman, editors, *DIALM-POMC*, pages 35–44. ACM, 2008.

[15] M. M. Halldórsson, S. Holzer, and N. A. Lynch. A local broadcast layer for the SINR network model. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC 2015, Donostia-San Sebastián, Spain, July 21 - 23, 2015*, pages 129–138, 2015.

[16] M. M. Halldórsson and P. Mitra. Nearly optimal bounds for distributed wireless scheduling in the sinr model. In *ICALP (2)*, pages 625–636, 2011.

[17] M. M. Halldórsson and P. Mitra. Towards tight bounds for local broadcasting. In F. Kuhn and C. C. Newport, editors, *FOMC*, page 2. ACM, 2012.

[18] M. M. Halldórsson and T. Tonoyan. How well can graphs represent wireless interference? In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 635–644, 2015.

[19] N. Hobbs, Y. Wang, Q.-S. Hua, D. Yu, and F. C. Lau. Deterministic distributed data aggregation under the sinr model. In *Theory and Applications of Models of Computation*, pages 385–399. Springer Berlin Heidelberg, 2012.

[20] T. Jurdzinski and D. R. Kowalski. Distributed backbone structure for algorithms in the sinr model of wireless networks. In M. K. Aguilera, editor, *DISC*, volume 7611 of *Lecture Notes in Computer Science*, pages 106–120. Springer, 2012.

[21] T. Jurdzinski and D. R. Kowalski. Distributed randomized broadcasting in wireless networks under the SINR model. In *Encyclopedia of Algorithms*, pages 577–580. 2016.

[22] T. Jurdzinski, D. R. Kowalski, M. Rozanski, and G. Stachowiak. Distributed randomized broadcasting in wireless networks under the sinr model. In *DISC*, pages 373–387, 2013.

[23] T. Jurdzinski, D. R. Kowalski, M. Rozanski, and G. Stachowiak. On the impact of geometry on ad hoc communication in wireless networks. In M. M. Halldórsson and S. Dolev, editors, *ACM Symposium on Principles of Distributed Computing, PODC '14, Paris, France, July 15-18, 2014*, pages 357–366. ACM, 2014.

[24] T. Jurdzinski, D. R. Kowalski, and G. Stachowiak. Distributed deterministic broadcasting in uniform-power ad hoc wireless networks. In L. Gasieniec and F. Wolter, editors, *FCT*, volume 8070 of *Lecture Notes in Computer Science*, pages 195–209. Springer, 2013.

[25] T. Jurdzinski, D. R. Kowalski, and G. Stachowiak. Distributed deterministic broadcasting in wireless networks of weak devices. In F. V. Fomin, R. Freivalds, M. Z. Kwiatkowska, and D. Peleg, editors, *ICALP (2)*, volume 7966 of *Lecture Notes in Computer Science*, pages 632–644. Springer, 2013.

[26] E. Kantor, Z. Lotker, M. Parter, and D. Peleg. The minimum principle of SINR: A useful discretization tool for wireless communication. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 330–349, 2015.

[27] E. Kantor, Z. Lotker, M. Parter, and D. Peleg. The topology of wireless communication. *J. ACM*, 62(5):37:1–37:32, Nov. 2015.

[28] T. Kesselheim. A constant-factor approximation for wireless capacity maximization with power control in the sinr model. In D. Randall, editor, *SODA*, pages 1549–1559. SIAM, 2011.

[29] T. Kesselheim. Dynamic packet scheduling in wireless networks. In D. Kowalski and A. Panconesi, editors, *PODC*, pages 281–290. ACM, 2012.

[30] T. Kesselheim and B. Vöcking. Distributed contention resolution in wireless networks. In N. A. Lynch and A. A. Shvartsman, editors, *DISC*, volume 6343 of *Lecture Notes in Computer Science*, pages 163–178. Springer, 2010.

[31] D. R. Kowalski. On selection problem in radio networks. In M. K. Aguilera and J. Aspnes, editors, *Proceedings of the Twenty-Fourth Annual ACM Symposium on Principles of Distributed Computing, PODC 2005, Las Vegas, NV, USA, July 17-20, 2005*, pages 158–166. ACM, 2005.

[32] D. R. Kowalski, W. K. M. Jr., and S. Vaya. Deterministic backbone creation in an SINR network without knowledge of location. *CoRR*, abs/1702.02460, 2017.

[33] D. R. Kowalski and A. Pelc. Broadcasting in undirected ad hoc radio networks. In *Proceedings of the Twenty-second Annual Symposium on Principles of Distributed Computing*, PODC '03, pages 73–82, New York, NY, USA, 2003. ACM.

[34] D. R. Kowalski and A. Pelc. Faster deterministic broadcasting in ad hoc radio networks. *SIAM J. Discrete Math.*, 18(2):332–346, 2004.

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

[36] E. Kushilevitz and Y. Mansour. An $\Omega(DLog(N/D))$ Lower Bound for Broadcast in Radio Networks. In *Proceedings of the Twelfth Annual ACM Symposium on Principles of Distributed Computing*, PODC '93, pages 65–74, New York, NY, USA, 1993. ACM.

[37] T. Moscibroda and R. Wattenhofer. The complexity of connectivity in wireless networks. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 23-29 April 2006, Barcelona, Catalunya, Spain*, 2006.

[38] D. Yu, Q. Hua, Y. Wang, and F. C. M. Lau. An o(log n) distributed approximation algorithm for local broadcasting in unstructured wireless networks. In *IEEE 8th International Conference on Distributed Computing in Sensor Systems, DCOSS 2012, Hangzhou, China, 16-18 May, 2012*, pages 132–139, 2012.

Appendix

A Extensions and conclusions

Optimizing Size of the Local Memory. In naive implementation of ProximityGraphConstruction procedure (see Algorithm 9) the local memory in each node is large. Nodes have to perform lookups in the wss, and a naive approach to this is that each node holds the whole selector, which is of size $\Theta(N \log N)$. We can reduce the size of local memory to $O(\log^2 N)$ in the following way. Each node holds only its pattern of transmissions, which results from the witnessed strong selector; its size is $O(\log N)$. In the exchange phase of Algorithm 9 each node sends its pattern of transmissions (a bit string of length $O(\log N)$). Thus, in edge removal phase each node has at most $O(\log N)$ "candidates" for neighbors and their transmission patterns, which is sufficient to perform edge removal. Thus, the total size of internal memory is $O(\log^2 N)$.

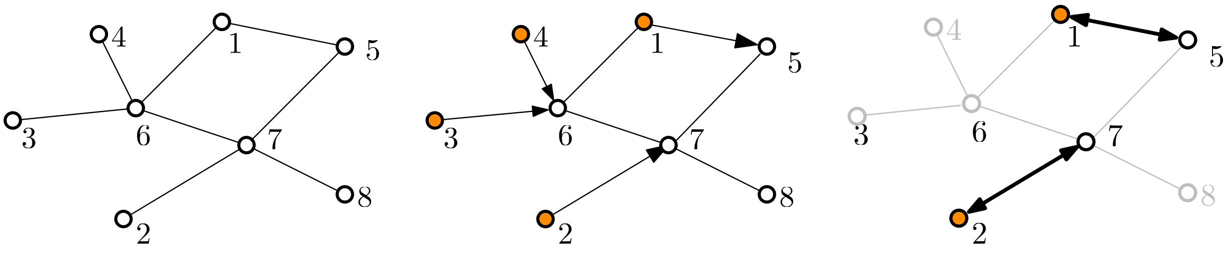
The amount of memory needed in TokenTraversal (see Algorithm 4) is $O(n \log N)$ bits due to the fact that a node can have $O(n)$ children. A node learns its children by FullSelection (Algorithm 5) in a sequential manner, one by one. This allows to form a kind of linked list of the children, such that each node remembers only the ID of the next node in the chain, reducing the amount of required memory to $O(\log N)$.

Acknowledged vs non-acknowledged broadcast. The algorithm presented in the paper works under the acknowledged broadcast model. (That is, for the broadcast problem, we require that an algorithm ends its execution when all nodes are aware of the fact that the broadcast message is delivered to the whole network.) In the case of non-acknowledged broadcast (i.e., an algorithm ends just at the moment of delivery of the broadcast message to all nodes) we can use a standard doubling technique, i.e., execute the algorithm for $n = 1, 2, 4, \ldots$ obtaining $O(n \log N)$ round complexity. In this way we get rid of the additive $\log^2 N$ term which might be $\omega(n \log N)$ if N is super-exponential wrt n.

Our results could be easily generalized to the bounded-growth metric space with the same asymptotic complexity bounds.

Conclusions. We presented a novel implementation of token traversal in weakly-connected ad hoc wireless networks under the uniform-power SINR model, leading to asymptotically optimal deterministic spanning tree and broadcast algorithm (see also the lower bound proved in this work) and nearly optimal randomized solution improving the one in [12]. The class of weakly-connected networks is the widest possible class, therefore our solution is on one hand most general, but on the other hand there are networks on which ad hoc communication is expensive by itself. 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.

B Auxiliary figures

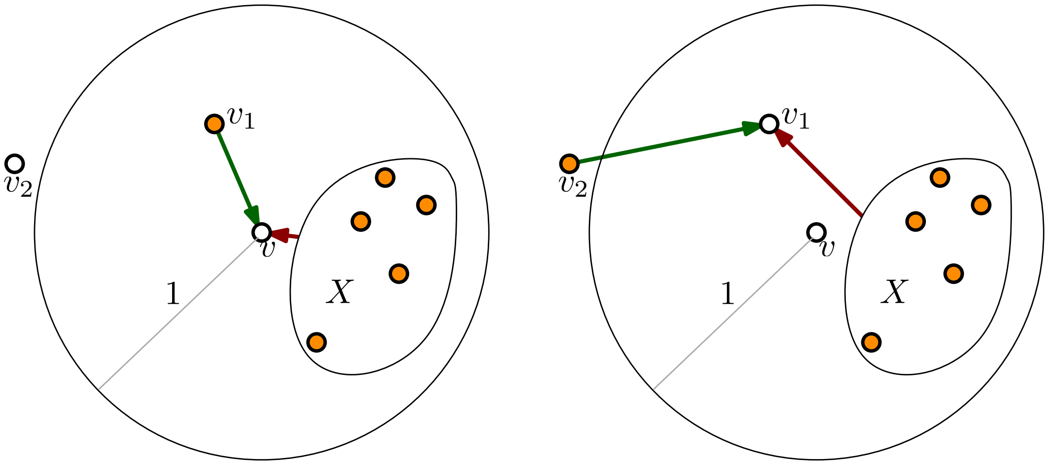


(a) A proximity graph H, with nodes labeled by their identifiers.

(b) Orange (shaded) nodes represent local minima. Each arrow goes from a local minimum v to $\min_v$.

(c) A local minimum v survives if v is the smallest neighbor of $\min_v$.

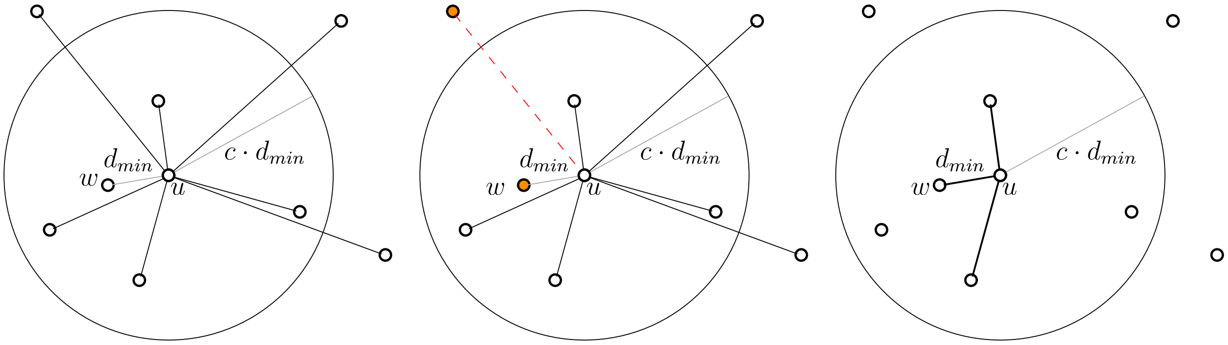
Figure 3: An illustration of the handshake procedure (Algorithm 2).



(a) Round 1

(b) Round 2

Figure 1: If $|X| \geq c_{\alpha,\beta}$, then interference from X prevents reception of the message from v_1 at v in Round 1 (part (a)) or reception of the message from v_2 at v_1 in Round 2 (part (b)). Otherwise, if $|X| < c_{\alpha,\beta}$, then v receives messages from each node from X in the execution of the $(N, c_{\alpha,\beta})$-ssf in step 5.



(a) Exchange phase: edges connect v with elements of U_v. Some of them can be "removed" in Edge removal phase.

(b) Edge removal phase: If v hears some node, it can "remove" all others that transmitted in that round.

(c) A possible result of the proximity graph construction. Only a pair of closest nodes is guaranteed to be connected by an edge.

Figure 2: An illustration of the proximity graph construction (Algorithm 9). The constant c corresponds to κ_0 from Lemma 4 and $d_{\min} < 1/2$ is the smallest distance between nodes in the whole set X.

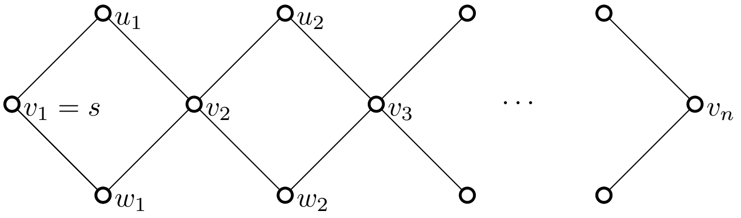
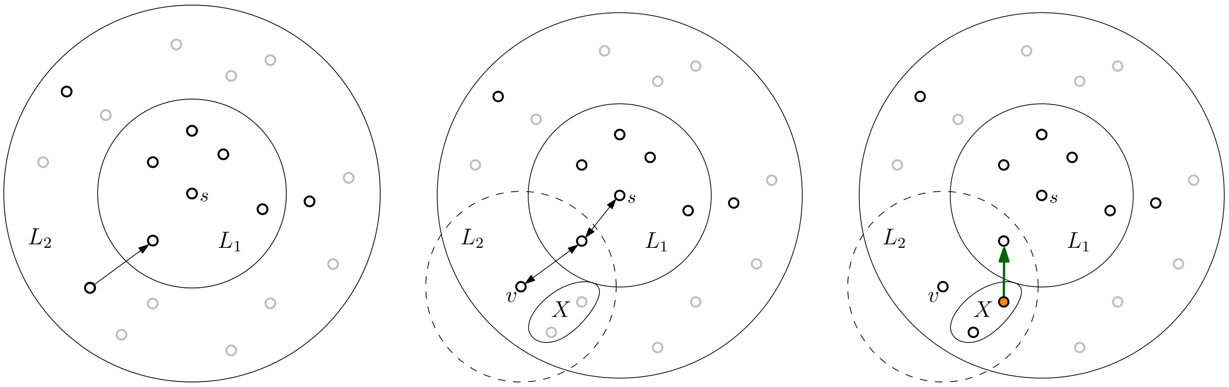


Figure 5: A chain-network used in $\Omega(n \log N)$ lower bound. Each link is of length 1.



(a) After sufficient reduction of density of $L \subseteq L_2$, some nodes have possibility to transmit uniquely in L. They are heard by their parents in L_1 which pass the information to the source.

(b) When the source receives a message from the parent of a node $v \in X$, it chooses v as the new leader, informs about it by sending a message through its parent $p(v)$. All not-yet-selected elements of L_2 within distance 1 to the new leader v are handled by FullSelection(v, X).

(c) Each node in X transmits uniquely during FullSelection(v, X) , on a distance 1, so it can be heard by its parent in L_1.

Figure 4: An illustration of the leader election and handling the second layer.

C Proofs of technical lemmas

Proof of Lemma 2

Proof. The proof is by probabilistic method. Let S_i be a set obtained by including each $v \in [N]$ with probability $1/k$. Moreover, for each i and v, choose whether $v \in S_i$ independently of other choices. There are

$$r = \binom{N}{k} \cdot k \cdot (N - k) \tag{2}$$

possible tuples (X, x, y) such that $|X| = k, x \in X, y \notin X$ and $X \subset [N]$, $y \in [N] \setminus X$, $x \in X$. For each i and a tuple (X, x, y), the event that $X \cap S_i = \{x\}$ and $y \in S_i$ occurs with probability

$$p = \frac{1}{k} \cdot \left(1 - \frac{1}{k}\right)^{k-1} \cdot \frac{1}{k} = \Omega(1/k^2). \tag{3}$$

The sequence $S_1, S_2, \ldots, S_m$ is a (N, k)-wss when, for each tuple (X, x, y) as above, the event $X \cap S_i = \{x\}$ and $y \in S_i$ occurs for some index $i \leq m$. The probability that this event does not occur for a tuple (X, x, y) for all indices $i \leq m$ is at most $(1 - p)^m$. The probability that there exists a tuple (X, x, y) for which the event $X \cap S_i = \{x\}, y \in S_i$ does not occur for all $i \leq m$ is smaller or equal to

$$r(1-p)^m \leq re^{-mp} = \binom{N}{k} k(N-k)e^{-mp} \leq N^{k+2}e^{-mp} = e^{\Theta(k \log N) - mp},$$

where r and p are defined in (2) and (3) respectively. By choosing large enough $m = \Theta(k^3 \log N)$, we get a nonzero probability of obtaining (N, k)-wss. Thus, such a selector of size $m = \Theta(k^3 \log N)$ exists. □

Proof of Lemma 3

Proof. Let $G_k = \{(X, x, y) : X \subseteq [N], |X| = k, x \in X, y \notin X\}$. We say that $S \subseteq [N]$ satisfies (X, x, y) iff $X \cap S = \{x\}$ and $y \in S$. Observe that $\mathcal{S}$ is a (N, k)-wss if and only if, for each $(X, x, y) \in G_k$, there exists $S \in \mathcal{S}$ that *satisfies* (X, x, y) that is $S \cap X = \{x\}$ and $y \in S$. The fact we exploit in the construction is that, for a fixed $\tilde{G} \subseteq G_k$, a random set $S_i \subseteq [N]$ constructed as in the proof of Lemma 2 (where each element of $[N]$ is put to the set S_i with probability $1/k$), *satisfies* a fraction of $\Omega(1/k^2)$ of triples in $\tilde{G}$ in expectation.

We construct the sets of the selector iteratively. Initially, $\mathcal{S} = \emptyset, \tilde{G} = G_k, i = 0$, and while $\tilde{G} \neq \emptyset$ we perform the following routine for consecutive is:

- $S_i \leftarrow \emptyset$
- For $z = 1, 2, \ldots, N$:
  - Let $Y(z)$ be the expected number of triples in $\tilde{G}$ satisfied by $S_i \cup \{z\}$, and $N(z)$ be the expected number of triples in $\tilde{G}$ satisfied by S_i, where the expected value is taken over random choices, where each element of $[z+1, N]$ is independently added to S_i with probability $1/k$.
  - If $Y(z) \geq N(z)$ then $S_i \leftarrow S_i \cup \{z\}$.
- $\mathcal{S} \leftarrow \mathcal{S} \cup \{S_i\}$, $\tilde{G} \leftarrow \tilde{G} \setminus \{$triples satisfied by $S_i\}$
- $i \leftarrow i + 1$

Since $\max\{Y(z), N(z)\} \geq \mathbb{E}[X] = \Omega(1/k^2)|\tilde{G}|$, where X is the expected number of triples from $\tilde{G}$ satisfied by a random set (cf. Proof of Lemma 2), after l repetitions of the above procedure, we have no more than $(1 - O(1/k^2))^l |G_k|$ triples (in $\tilde{G}$) unsatisfied by all sets $S_1, \ldots, S_l$. Since $|G_k| = \binom{N}{k} k(N-k) = O(N^{k+2})$, there are no unsatisfied triples in $\tilde{G}$ after $l = O(k^3 \log N)$ repetitions of the above procedure, and we obtain an (N, k)-wss $\mathcal{S} = (S_1, \ldots, S_l)$. The total time complexity of the algorithm is $O(k^4 \log N \cdot f(N))$, where $f(N)$ is a function determining time needed to compute $Y(z)$ and $N(z)$. In order to confirm that $Y(z)$ and $N(z)$ might be computed in time $N^{O(k)}$, we observe that probability that a given triple (X, x, y) is satisfied by S_i or $S_i \cup \{z\}$ can be computed in time $N^{O(1)}$. Given the triple (X, x, y) and a fixed value of z, the probability that the triple (X, x, y) will be satisfied by S_i depends solely on the facts whether

- x, y are in S_i at the current stage, whether
- $x < z, x = z$ or $x > z$,
- $y < z, y = z$ or $y > z$
- the size of $X \cap S_i$ is 0, 1 or larger than 1.

Thus, time complexity is $N^{O(k)}$ as claimed. For more details regarding this technique, we refer to [9]. □

Proof of Lemma 4

Proof. Let $d \leftarrow d(u, v)$. As (u, v) is a closest pair, $d(x, y) \geq d$ for each $x \neq y$, $x, y \in A$. Let A_i be the elements of A located in $\mathcal{B}(v, (i+1)d) \setminus \mathcal{B}(v, id)$ for natural i. The fact that the distance between each pair of nodes is at least d implies that interference at v from the elements of A_i is $O(i \cdot \frac{1}{(id)^\alpha})$, since $|A_i| = O(i)$ and $d(v, x) \geq id$ for each $x \in A_i$. Let κ_0 be a fixed constant. Then, interference at v from all nodes outside of $B(v, \kappa_0 d)$ is

$$O\left(\sum_{i \geq \kappa_0} i \cdot \frac{1}{(id)^\alpha}\right) = O\left(\frac{1}{d^\alpha} \sum_{i \geq \kappa_0} 1/i^{\alpha-1}\right).$$

The assumption $\alpha > 2$ implies that $\sum_{i \geq 0} 1/i^{\alpha-1} = O(1)$ and therefore, for each constant c we can guarantee that $\frac{1}{d^\alpha} \cdot \sum_{i \geq \kappa_0} 1/i^{\alpha-1} < c$ by choosing large enough κ_0. As $d(u, v) \leq 1/2$, the lemma holds for κ_0 adjusted to such c that

$$\frac{\frac{1}{(1/2)^\alpha}}{\mathcal{N} + c} \geq \beta.$$

□

Proof of Lemma 5

Proof. If X is non-empty then either v receives a message during the execution of $(N, c_{\alpha,\beta})$-ssf if $|X| < c_{\alpha,\beta}$ (step 5) or the interference from X prevents at least one of the transmissions $v_1 \to v$, $v_2 \to v_1$ (Rounds 1 and 2). (The latter fact follows from the choice of the constant $c_{\alpha,\beta}$ and the assumption $X \subseteq N_v$ which implies that X is included in $\mathcal{B}(v, 2)$ and $\mathcal{B}(v_1, 2)$.) In both cases the algorithm returns false: in line 6 in the former case and in line 9 in the latter case.

Otherwise, if X is empty, the transmissions in Rounds 1, 2, 3 succeed and v reports the emptiness of X.

The complexity of $(N, c_{\alpha,\beta})$-ssf dominates the complexity of the routine, which is $O(c^2_{\alpha,\beta} \log N) = O(\log N)$ by Lemma 1. □

Proof of Lemma 6

Proof. The result follows from the fact that a station can hear only as many different stations as the number of rounds it listened. Thus the value of p in the algorithm is at most as large as the length of exchange phase, that is $|T|$. □

Proof of Lemma 7 First, we provide a pseudo-code of the considered algorithm:

Algorithm 9 ProximityGraphConstruction(v)

```
$E_v \leftarrow \emptyset$
$\mathbf{S} \leftarrow (N, \kappa)$-wss common to all nodes.        ▷ $\kappa$ is the constant from Corollary 1
**Exchange phase.**
Execute $\mathbf{S}$ with message containing ID($v$).
Let $U_v$ be the nodes which successfully delivered a message to $v$ during the exchange phase.
**Edge removal phase.**
$C_v \leftarrow U_v$
**for each** $u, w \in U_v$ **do**
    **if** in some round $u$ and $w$ transmitted **and** $v$ heard $u$ **then**        ▷ lookup in the schedule $\mathbf{S}$
        $C_v \leftarrow C_v \setminus \{w\}$
**Confirmation phase.**
**if** $|C_v| > \kappa$ **then**
    $C_v \leftarrow \emptyset$
**for each** $u \in C_v$ **do**
    Execute $\mathbf{S}$ with $\langle v, u \rangle$ as the message.
**for each** message $\langle w, v \rangle$ received during the confirmation phase **do**
    **if** $w \in C_v$ **then**
        $E_v \leftarrow E_v \cup \{w\}$
```

Below, the formal proof follows.

Proof. Let $X \subseteq V$ be the subset of nodes executing the procedure. First, we need to assure that each closest pair in X is connected by an edge in H. Let $u, w \in X$ be a closest pair. We show that $u \in E_w$ and $w \in E_u$ after an execution of ProximityGraphConstruction on X.

By Corollary 1 we know that if u is the only transmitter among κ nodes closest to w (including w) then w receives the message. By the definition of the schedule $\mathbf{S}$ which is a (N, κ)-wss we know that a round satisfying this condition exists in $\mathbf{S}$. Thus, after Exchange phase, u and w have each other in the set of candidates U_u/U_w (see Fig. 2(a)). Since u, w is a closest pair, it is impossible that u receives a message from $x \neq w$ in a round in which w transmits a message (since $\beta > 1$, only the message from the closest transmitter can be received). Thus, w (u, resp.) belongs to C_u (C_w, resp.) after Edge removal phase.

A node can also purge the candidate set C_v during the confirmation phase, if the set of candidates contains more than κ nodes. However, as $\mathbf{S}$ is (N, κ)-wss, if $u, w \in X$ is a closest pair then C_u and C_w are of size at most κ. Indeed, let U denote the set of κ nodes closest to u (including u). Then, for any node $u_{\text{far}} \notin U$ which is not among κ closest nodes to u, there is a round in $\mathbf{S}$ in which w transmits uniquely in U and also u_{far} transmits by the witnessed strong selection property of $\mathbf{S}$. Thus, u_{far} is eliminated from set C_u in Edge removal phase, as well as any other node not in U. So, the set of candidates C_u for u is a subset of U, thus its size is at most κ (see Fig. 2(b)).

It is clear that the degree of resulting graph is at most $\kappa = O(1)$. The round complexity of the procedure is at most $(\kappa + 1)|\mathbf{S}| = O(\log N)$. □

Proof of Lemma 8

Proof. Assume that there exists a closest pair $u, w \in X$. By Lemma 7, the set of edges of $H(X)$ is non-empty in this case.

First, we show that $|Y| > 1$. Let u be the node with the smallest ID among nodes with non-zero degree in $H(X)$. Let $w = \min_u$. Then, $\min_w = u$ and u survives as the conditions checked in step 5 is satisfied for u.

Next, observe that y survives iff the node $x = \min_y$ satisfies $y = \min_x$ and x does not survive. Thus, the number of nodes from X which do not survive is not smaller than Y and therefore $|Y| \leq |X|/2$. Moreover $x = p(y)$ belongs to $X \setminus Y$ when y survives.

The round complexity of Handshake(X) is $O(\log N)$ which follows from the optimal size of witnessed strong selectors and complexity of ProximityGraphConstruction (Lemma 7). □

Proof of Lemma 9

Proof. First, assume that X is not empty. Let X_i be the value of X after the ith repetition of the repeat-loop, $X_0 = X$. If $|X_i| \geq \chi(1/2, 1)$ then, by Lemma 8, $1 \leq |X_{i+1}| \leq |X_i|/2$. If $|X_i| < \chi(1/2, 1)$ then all nodes transmit separately in line 6 and are heard by v. Thus, the node w is returned by the algorithm after r repetitions of the repeat-loop such that $|P(w)| = r$. Thus, item 1. of the lemma holds.

Second, assume that $X = \varnothing$. If $X = \varnothing$ and $v \in L_0 \cup L_1$ then v cannot run effective test of emptiness for X, thus it executes $\log N$ iterations of the Handshake and learns that X is empty. Each execution of Handshake takes $O(\log N)$ rounds, thus the total round complexity is $O(\log^2 N)$. Otherwise, if $X = \varnothing$ and $v \notin L_0 \cup L_1$ then v learns that X is empty with EmptinessTest by Lemma 5. □

Proof of Proposition 1

Proof. Lemma 8 implies that the size of L is reduced at least twice and it remains greater than zero after each execution of Handshake(L), as long as there is a closest pair in L. As L is included in a ball of radius 2, a closest pair in L exists for sure, provided $|L| > \chi(1/2, 2)$. Thus, after $i = O(\log n)$ repetitions of Handshake(L), we have $0 < |L| \leq \chi(1/2, 2)$. Then, each element of still non-empty L transmits uniquely in an execution of (N, k)-ssf (see line 4 of Alg. 8).

When u transmits uniquely in L_2, it is heard by its parent, which transmits a leader-proposal message, which is heard by the source. After reception of the leader-proposal message, s transmits back the leader-elect message (and sets the local variable elected to true) assuring that the unique leader is chosen. □

Proof of Lemma 11

Proof. Proposition 1 guarantees that at most $\chi(1, 2)$ leaders are elected in the main loop, such that each $v \in L_2$ is either a leader or a neighbor of some leader. Every node non-leader $u \in L_2$ is selected by some leader during FullSelection and, by Lemma 10, u transmits uniquely in L_2, and is heard by parent(u) $\in L_1$. Each node $u \in L_2$ is handled only once by some leader $v \in L_2$ during FullSelection and, for each leader, the procedure takes $O(|X| \log N + \log^2 N)$ rounds, where X is the set of nodes handled by v. Thus, the total round complexity of HandleSecondLayer is $O(\chi(1, 2) \cdot \log^2 N + |L_2| \log N)$ which implies the claimed result, since $\chi(1, 2) = O(1)$ and $|L_2| = O(n)$. □