8-1-2008

Computing the k-hop neighborhoods in wireless networks locally

Iyad A. Kanj
DePaul University

Andreas Wiese
Technische Universitat Berlin

Fenghui Zhang
Texas A & M University - College Station

Computing the k-hop neighborhoods in wireless networks locally

Iyad A. Kanj* Andreas Wiese† Fenghui Zhang‡

Abstract

A *k-local* distributed algorithm (k is a natural number) is a distributed algorithm in which the computation at every point/device in the distributed system modeled as a graph depends solely on the initial states of the points that are at most k hops away from the point. A distributed algorithm is *local* if it is k-local for some fixed natural number k. Local distributed algorithms are very important, especially for applications in ad-hoc sensor and wireless networks, since such algorithms are naturally scalable, robust, and fault tolerant. Clearly, an essential component of any k-local distributed algorithm is computing the k-hop neighborhoods of the points in the graph.

In this paper we show that, given a wireless network modeled as a unit ball graph or as a quasi unit ball graph U on n points in the 3-dimensional (3-D) Euclidean space, and a natural number k, there exists a local distributed algorithm that computes the k-hop neighborhoods of the points in U such that the total number of messages sent by all the points in U at the end of the algorithm is $O(n)$, and where each message has length $O(\lg n)$ bits. Clearly, this algorithm is optimal.

We note that the same results can be projected to unit disk graphs and quasi unit disk graphs in the Euclidean plane. Moreover, our techniques, and hence our results, can be generalized in a straightforward manner to unit ball graphs in ℓ-dimensional spaces, where $\ell > 3$ is an integer.

1 Introduction

Two devices in wireless ad-hoc and sensor networks can, in principle, communicate if they are in each other's transmission range. Therefore, when studying such networks, it is natural to embed them in a Euclidean metric space. A common simple embedding assumes that the space is two dimensional, and that the transmission range of all devices is the same. In that case, the network is modeled as a *Unit Disk Graph*, abbreviated UDG, defined as the subgraph of the *Euclidean graph*—the complete graph on the same set of points—consisting of edges of length at most 1 unit. In practice, however, the UDG model may be too idealistic for wireless sensor networks. Fading signal strength [4, 8], antenna design issues, inaccurate position estimation, physical obstructions and other issues may cause the transmission range of devices to deviate from unit disks. For this reason, a more practical network model, which is a generalization of the UDG model, has been proposed. The *quasi unit disk graph*, shortly quasi-UDG, has been proposed to capture the non-uniform characteristics of wireless sensor networks [2]. Let $0 \leq r \leq 1$ be a constant. A *quasi unit disk graph*, abbreviated quasi-UDG henceforth, with parameter r in the Euclidean plane is defined as follows: for any two points p and q in the graph, pq is an edge if

*School of Computing, DePaul University, 243 S. Wabash Avenue, Chicago, IL 60604-2301, USA. Email: ikanj@cs.depaul.edu.

†Institute für Mathematik, Technische Universität Berlin, Germany. Email: wiese@math.tu-berlin.de.

‡Department of Computer Science, Texas A&M University, College Station, TX 77843-3112, USA. Email: fhzhang@cs.tamu.edu.

$|pq| \leq r$, and pq is not an edge in the graph if $|pq| > 1$. If $r < |pq| \leq 1$ then pq may or may not be an edge in the graph; it is usually assumed that an adversary decides the placement of such edges. The quasi-UDG model was first studied in [2], and further developed in [5].

The 2-dimensional embedability of the network, assumed by both UDGs and quasi-UDGs, is often too simplistic and inappropriate. For example, sensors maybe spread throughout a high-rise (e.g., city), or in space (e.g., battlefield, navigation). In such cases a 3-dimensional embeding/model can be used instead. For an integer $\ell \geq 3$, the *Unit Ball Graph* abbreviated UBG, is a straightforward generalization of the UDG to Euclidean spaces of dimension ℓ: two points in a UBG are connected if and only if their Euclidean distance in the ℓ-dimensional Euclidean space is at most 1 unit. Similarly, the ℓ-dimensional *quasi Unit Ball Graph*, abbreviated quasi-UBG henceforth, is a straightforward generalization of a quasi-UDG.

Scalability, fault tolerance, efficiency, and robustness, are all key issues in wireless ad-hoc and sensor networks whose devices have limited computational (battery) power, high mobility, and usually lack the centralized coordination. Distributed algorithms that use global propagation of information, or simulate centralized algorithms, are not typically sensitive to these issues. Therefore, the algorithms designed for such networks should be simple, efficient both in terms of time and space, scalable, and robust to environment changes.

Most of the the above requirements on the distributed algorithms can be captured by the notion of *locality*, as defined by Linial [6] and Peleg [7]. Assuming that the distributed system is modeled as a graph, a distributed algorithm for such a model is said to be *k-local* if, "intuitively", the computation at each point of the graph depends solely on the initial state of the points at distance (number of edges) at most k from the point (i.e., within k hops from the point). Wattenhofer [10] formalizes this notion as follows: a distributed algorithm is *k-local* if it runs in at most k synchronous communication rounds for some natural number k. Efficient distributed algorithms that are k-local are naturally scalable and robust to changes because "change" can be handled locally by such algorithms. Therefore, it is natural to seek local distributed algorithms when solving fundamental problems in distributed systems.

A typical k-local distributed algorithm needs to compute the k-hop neighborhoods of every point in the graph. Obviously, this can be accomplished by running k communication rounds, where in each round every point in the graph broadcasts all the information (IDs and coordinates) it received in the preceding communication round; in the first round this information consists only of the coordinates of the point itself. However, in the worst case, the number of messages sent by this brute-force algorithm can be as large as $\Omega(n^2)$, where n is the number of points in the graph. This can be seen, for example, in the case when the graph is the complete graph on n points; in this case each point in the graph will send $O(n)$ messages. Calinescu [3] developed a distributed algorithm for computing the 2-hop neighborhoods in a UDG on n points using $O(n)$ messages; however, his algorithm is not local and uses a non-local distributed algorithm for computing a connected dominating set in a UDG, given in [1, 9].

The contribution of this paper is in presenting a local distributed algorithm for computing the k-hop neighborhoods of the points in a UBG or a quasi-UBG U in the 3-D Euclidean space, where $k \geq 0$ is a given integer constant, in which the total number of messages sent by all points in U is $O(n)$, and where the length of a message is $O(\lg n)$ bits. Clearly, this algorithm is optimal. Our techniques can be projected in a straightforward manner to the Euclidean plan, and hence, the same results hold true for UDGs and quasi-UDGs. Moreover, our techniques, and hence our results, can be generalized in a straightforward manner to UBGs and quasi-UBGs in ℓ-dimensional spaces, where $\ell > 3$ is an integer.

When $k = 1$, the brute-force algorithm reduces to an algorithm in which every point broadcasts its coordinates, and hence the total number of messages sent by all the points at the end of this algorithm is clearly $O(n)$. Moreover, if $k = 0$, then clearly no messages need to be sent

in this case. Therefore, we will focus in this paper on the cases when $k \geq 2$.

2 Preliminaries

Given a set of points $\mathcal{P}$ in the 3-D Euclidean space, the Euclidean graph $\mathcal{E}$ on $\mathcal{P}$ is defined to be the complete graph whose point-set is $\mathcal{P}$. The *unit ball graph*, shortly *UBG*, U on $\mathcal{P}$ is the subgraph of $\mathcal{E}$ with the same point-set as $\mathcal{E}$, and such that pq is an edge of U if and only if $|pq| \leq 1$, where $|pq|$ is the Euclidean length of edge pq.

Let $0 < r \leq 1$ be a constant. The *quasi-UBG* on $\mathcal{P}$ with parameter r, is the subgraph U_r of U with the same point-set as U, and such that for any two points p and q in U_r: if $|pq| \leq r$ then $|pq|$ is an edge of U_r, if $r < |pq| \leq 1$ then pq may or may not be an edge of U_r, and if $|pq| > 1$ then pq is not an edge of U_r. Clearly, a UBG is a quasi-UBG with parameter $r = 1$.

Throughout this paper we will always assume that the UBG or quasi-UBG U is connected.

For a subgraph $H \subseteq \mathcal{E}$, we denote by $V(H)$ and $E(H)$ the set of vertices and the set of edges, respectively, of H. The length of a path P in a subgraph $H \subseteq \mathcal{E}$, denoted $|P|$, is the number of edges in P. A point q is said to be an *i-neighbor* of a point p in a subgraph $H \subseteq \mathcal{E}$, if there exists a path P from p to q in H satisfying $|P| \leq i$.

For a set of vertices S in U and a point $p \in S$, denote by $deg_S(p)$ the degree of point p in the subgraph of U induced by S, and denote by Δ_S the value $max\{deg_S(p) \mid p \in S\}$; we let $deq(p)$ denote $deg_U(p)$. For a point $p \in U$, denote by $N_k[p]$ the *k-hop neighborhood* of p in U, i.e., the set of k-hop neighbors of p in U. For a vertex p in a subgraph H of U, denote by $N_k^H[p]$ the set of k-neighbors of p in H.

The goal of this paper is to show how the k-hop neighborhoods of the points in U can be computed efficiently. Our efficiency metric here is the total number of messages sent by the points of U when the algorithm terminates, as well as the total number of messages sent by each point in U. We will assume, as is commonly the case in the relevant literature, that the coordinates and ID of any point in U can be encoded using $O(\lg n)$ bits, where $n = V(U)$. We will assume that the communication model is the *broadcast model*: when a point p in U sends a message, the message is received by all its neighbors.

We start by considering the case of UBGs, and then we will show how the results can be extended to quasi-UBGs.

3 The local distributed algorithm for UBGs

The algorithm presented in this paper relies on a local distributed algorithm for constructing connected dominating sets of UBGs, similar to the algorithm for UDGs given in [11]. We summarize this algorithm next and prove some of its properties that are essential to the results in this paper. We will write *CDS* for connected dominating set.

The algorithm on UDGs in [11] starts by tiling the plane with hexagons, each of diameter 1. Therefore, the points in each hexagon form a clique in U. Then, for any two adjacent[1] (or identical) hexagons, the two endpoints of the shortest edge connecting the two hexagons are added to a set D. It was proved in [11] that D is a connected (that is, the subgraph of U induced by D is connected) dominating set of U whose size is at most 216 times the size of a minimum CDS of U.

For UBGs, we first pack the space with cubes of diameter 1 (the demision is then $1/\sqrt{3}$), among all the edges connecting two different cubes, we put the endpoints of the shortest such edge into D. Assuming the graph is connected, the only case when a cube is isolated is when the

[1]Some point in the first hexagon is adjacent to some point in the second hexagon.

whole graph is within a single cube. In that case we pick an arbitrary point in the cube and put it into D. We will refer to this algorithm by the **Connected Dominating Set Algorithm**, shortly the **CDS-Algorithm**.

Lemma 3.1. *Let D be the set of points constructed by the* **CDS-Algorithm** *on the UBG U. Then D is a CDS and:*

(a) The number of points in D contained in any bounded region (of constant volume) of the 3-D Euclidean space is $O(1)$.

(b) Δ_D is a constant; that is, the number of neighbors in D of any point in U is $O(1)$.

(c) For any two points p and q that are i-neighbors in U, there exists a point $u \in D$ in the same cube as p, and there exists a point $v \in D$ in the same cube as q, such that u and v are $(2i-1)$-neighbors in the subgraph of U induced by D.

(d) The set D can be constructed by a 2-local distributed algorithm in which the number of messages sent by any point $p \in U$ during the execution of the algorithm is $O(1)$. Consequently, the total number of messages sent by all points in U is $O(n)$.

Proof. It is straightforward that D is a dominating set of U. This is because any point p in U resides in some cube, and at least one point from each cube is in D. Since within each cube all the points form a clique, p is either in D or is dominated by some point in D. The connectedness of the subgraph induced by the points in D follows immediately from part (c), which will be proved below.

To prove part (a), observe that any cube in the tiling has a constant number of adjacent cubes. Therefore, the number of points in D contained in any cube is $O(1)$. Since the number of cubes intersecting any bounded region in the space is $O(1)$, part (a) follows.

Part (b) follows from part (a) since the neighbors of any point in U are contained within a bounded region of the space (sphere/ball centered at the point and of radius 1).

To prove part (c), let p and q be two i-neighbors in U, and let $P = (p = p_0, p_1, \ldots, p_r = q)$ be a path between p and q in U, where $r \leq i$. Observe that, since U is connected, it follows from the construction of D that for every point $p \in U$ there exists a point x in D such that x and p belong to the same cube. For every point p_j on P, $j = 0, \ldots, r$, we will correspond a point u_j in D that dominates p_j, and such that u_0 and u_1 are 1-neighbors in $U(D)$, and $u_{\ell-1}$ and u_ℓ are 2-neighbors in $U(D)$, for $\ell = 2, \ldots, r$. Note that this will prove part (c). If p_0 and p_1 belong to the same cube, let $u \in D$ be a point in the same cube as p_0 and p_1. We let $u_0 = u_1 = u$. If p_0 and p_1 belong to two different cubes, then from the construction of D, there exists a point $x \in D$ in the same cube as p_0, and $y \in D$ in the same cube as p_1, such that xy is the shortest edge connecting the two cubes. We let $u_0 = x$ and $u_1 = y$. Observe that, in both cases, u_0 and u_1 are 1-neighbors in D. Suppose inductively that point u_ℓ has been defined for point p_ℓ, where $2 \leq \ell < r$, we define point $u_{\ell+1}$ as follows. If $p_{\ell+1}$ is in the same cube as p_ℓ, then we let $u_{\ell+1} = u_\ell$; otherwise, from the construction of D, there exists a point x in the same cube as p_ℓ, and a point y in the same cube as $p_{\ell+1}$, such that xy is the shortest edge connecting the two cubes. We let $u_{\ell+1} = y$. Observe that since x and u_ℓ are in the same cube, they are 1-neighbors in D, and hence, u_ℓ and $u_{\ell+1}$ are two neighbors in D. By letting $u = u_0$ and $v = v_r$, part (c) follows.

For part (d), note that the algorithm that constructs D is 2-local. This can be seen as follows. First, every point in U broadcasts its coordinates. In this step every point p in U broadcasts $O(1)$ messages. Then every point p in cube $\mathcal{H}$ of U, and for each cube $\mathcal{H}'$ adjacent to $\mathcal{H}$, p computes the shortest incident edge connecting it to $\mathcal{H}'$ (if such an edge exists); there are $O(1)$ such cubes $\mathcal{H}'$, and hence the information about these shortest edges can be stored using $O(1)$

messages. Then every point broadcasts this information to its neighbors. By the same token, in this step every point $p \in U$ sends $O(1)$ messages. Now every point has all the information needed to compute the shortest edge connecting its cube to all adjacent cubes, and can decide whether it is an endpoint of such a shortest edge or not, and consequently, whether it belongs to D or not. Clearly, the above algorithm is 2-local because (it runs in 2 synchronous rounds) every point decides whether it belongs to D or not based solely on the information about its 2-hop neighbors. Therefore, from the above discussion, at the end of the algorithm every point p in U has sent $O(1)$ messages. It follows that the total number of messages sent by all points in U is $O(n)$.

This completes the proof. □

The algorithm we propose for computing the k-hop neighborhoods of the point in U is very simple. First, the **CDS-Algorithm** is applied to compute a connected dominating set D of U. After the execution of the **CDS-Algorithm**, we run $2k+1$ communications rounds. In the first round, every point in U broadcasts its coordinates; in the i-th round, where $2 \leq i \leq 2k+1$, every point in D broadcasts all the *new* information/coordinates that it received in round $i-1$. Formally, every point $p \in U$ executes the algorithm given in Figure 1. We will refer to this algorithm as the k-**hop Neighborhoods** algorithm.

Algorithm k-**hop Neighborhoods**

Round 1:

1. p broadcasts its coordinates;

Round i, $2 \leq i \leq 2k$:

1. if p is in D then p broadcasts all the information received in round $i-1$;

Figure 1: The k-hop Neighborhoods Algorithm.

Lemma 3.2. *The algorithm* k-**hop Neighborhoods** *is a* $(2k+1)$*-local distributed algorithm.*

Proof. The statement of the lemma follows immediately from the fact that the algorithm k-**hop Neighborhoods** runs in $2k+1$ communication rounds. □

Lemma 3.3. *After the execution of the algorithm* k-**hop Neighborhoods**, *every point in* U *has learned the coordinates of its* k*-hop neighbors.*

Proof. Let p be a point in U, and let q be a k-neighbor of p. It suffices to show that after the execution of the algorithm k-**hop Neighborhoods**, point p has learned the coordinates of point q.

By part (d) of Lemma 3.1, there exist points u and v in D such that u is a neighbor of p and v a neighbor of q, and such that u and v are $(2k-1)$-neighbors in D. Since the algorithm k-**hop Neighborhoods** runs in $2k+1$ rounds, after $2k$ rounds, the coordinates of q have "propagated" from point v to point u. In round $2k+1$, u broadcasts its information to its neighbors, including point p. It follows that point p will learn about the coordinates of q when the algorithm terminates. □

Lemma 3.4. *Let* p *be a point in* U*. Then the number of messages sent by* p *when the algorithm* k-**hop Neighborhoods** *terminates is* $O(|N_{2k+1}[p]|)$.

Proof. To analyze the number of messages sent by point p in the algorithm k-**hop Neighborhoods**, observe first that if p is a point in $U \setminus D$ then p only sends its coordinates in round 1.

Suppose now that $p \in D$. Since the algorithm k-**hop Neighborhoods** runs in $2k+1$ rounds, point p could only receive the coordinates of the points in $N_{2k+1}[p]$ when the algorithm terminates. Since every point broadcasts the same message at most once, the number of messages sent by point p after the termination of the algorithm is $O(|N_{2k+1}[p]|)$. □

Lemma 3.5. *The total number of messages sent by all points in U when the algorithm k-***hop Neighborhoods** *terminates is $O(n \cdot \min(c^{2k}, N_{avg}^{2k+1}))$, where c is a constant, n is the number of points in U, and N_{avg}^{2k+1} is the average size of the $(2k+1)$-neighborhoods of the points in U, that is, $N_{avg}^{2k+1} = (1/n) \cdot \sum_{p \in U} |N_{2k+1}[p]|$. In particular, for any integer constant k, the total number of messages sent by all points in U at the end of the algorithm is $O(n)$.*

Proof. Each message contains the ID and the coordinates of some point in U. Let p be a point in U, and let M_p be the message containing p's ID and coordinates. This message was sent once in round 1 by p. In the following rounds, only those points that are in $D_p = N_{2k} \cap D$ could send M_p. By part (b) of Lemma 3.1, every point in U has at most $c = O(1)$ neighbors in D. Therefore, the cardinality of D_p is bounded by c^{2k}. Since every point in D_p sends M_p at most once, it follows that the total number of times the message M_p was sent is $O(c^{2k})$. Consequently, the total number of messages sent by all points in U at the end of the algorithm is $O(nc^{2k})$. If k is fixed, then $O(nc^{2k}) = O(n)$.

On the other hand, by Lemma 3.4, the total number of messages sent by a point $p \in U$ after the algorithm terminates is $O(|N_{2k+1}[p]|)$. Therefore, the total number of messages sent by all points in U is $O(\sum_{p \in U} |N_{2k+1}[p]|) = O(n \cdot (1/n) \cdot (\sum_{p \in U} |N_{2k+1}[p]|)) = O(n \cdot N_{avg}^{2k+1})$.

It follows that the total number of messages sent is bounded by $O(n \cdot \min(c^{2k}, N_{avg}^{2k+1}))$. □

Theorem 3.6. *Let U be a connected UBG with n points and m edges, and let $k \geq 2$ be an integer constant. There exists a $(2k+1)$-local distributed algorithm that, for every point $p \in U$ computes precisely the set $N_k[p]$. The number of messages sent by any point $p \in U$ at the end of the algorithm is $O(|N_{2k+1}[p]|)$. The total number of messages sent by all points in U at the end of the algorithm is $O(n \cdot \min(c^{2k}, N_{avg}^{2k+1}))$, where c is a constant. In particular, for any integer constant k, the total number of messages sent by all points in U at the end of the algorithm is $O(n)$.*

Proof. Consider the algorithm k-**hop Neighborhoods**. By Lemma 3.3, every point $p \in U$ learns about its k-hop neighbors at the end of the algorithm. However, since the algorithm runs in $2k+1$ rounds, a point p may have also learned about the coordinates of its i-neighbors, where $k < i \leq 2k+1$. To compute the set $N_k[p]$ precisely, point p needs to identify the coordinates of the points that are not in $N_k[p]$ that it may have received. For this purpose, point p runs a breadth first search algorithm on the subgraph of U induced by the all points whose coordinates it has received at the end of the algorithm. Point p then identifies those points whose distance to it (length of a shortest path) is more than k, and discards them. Clearly, with this simple tweak of the algorithm k-**hop Neighborhoods**, every point in U will compute the set $N_k[p]$ precisely. Moreover, this additional step involves no communication with other points, and hence the communication complexity of the algorithm does not change.

The remaining facts in the statement of the theorem follow from the discussion and lemmas above. □

We conclude with the following theorem:

Theorem 3.7. *Let U be a UBG on n points, and let $k \geq 0$ be an integer constant. There exists a local distributed algorithm that computes the k-hop neighborhoods of all points in U such that the total number of messages sent by all points in U at the end of the algorithm is $O(n)$.*

Since UBGs are straightforward generalizations of UDGs in the Euclidean plane, the above theorem applies to UDGs as well.

4 Extension to quasi-UBGs

The key component of the local distributed algorithm described in the previous section is the construction of a CDS satisfying the properties stated in Lemma 3.1. For a given quasi-UBG U, similarly to the algorithm in [11], we can obtain a CDS satisfying all the conditions in Lemma 3.1:

Given a quasi-UBG U with parameter r, we modify the **CDS-Algorithm** in [11] slightly as follows. Instead of packing the space with cubes of dimension $1/\sqrt{3}$, we use cubes of dimension $\sqrt{r/3}$. The rest of the algorithm is kept the same. We shall call this modified algorithm the **QUBG-CDS-Algorithm**. Since we only reduced the dimension of the cubes to a smaller constant, it is not difficult to see that the arguments in the proof for Lemma 3.1 still hold true. Consequently, we have the following lemma:

Lemma 4.1. *Let D be the set of points constructed by the* **QUBG-CDS-Algorithm** *when applied to the quasi-UBG U. Then D is a CDS satisfying all the conditions stated in Lemma 3.1.*

Once we have constructed the CDS D with the desired properties, the **k-hop Neighborhoods** algorithm works exactly in the same way, only with larger constants in the message complexity. Therefore, we conclude with the following theorem:

Theorem 4.2. *Let U be a quasi-UBG on n points, and let $k \geq 0$ be an integer constant. There exists a local distributed algorithm that computes the k-hop neighborhoods of all points in U such that the total number of messages sent by all points in U at the end of the algorithm is $O(n)$.*

Since quasi-UBGs are straightforward generalizations of quasi-UDGs in the Euclidean plane, the above theorem applies to quasi-UDGs as well.

References

[1] Khaled M. Alzoubi, Peng-Jun Wan, and Ophir Frieder. Message-optimal connected dominating sets in mobile ad hoc networks. In *Proceedings of the 3rd ACM Interational Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 157–164, 2002.

[2] L. Barrière, P. Fraigniaud, and L. Narayanan. Robust position-based routing in wireless Ad Hoc networks with unstable transmission ranges. In *Proc. of the 5th Int. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 19–27. ACM, 2001.

[3] G. Calinescu. Computing 2-hop neighborhoods in Ad Hoc wireless networks. In *Proceedings of ADHOC-NOW*, volume 2865 of *Lecture Notes in Computer Science*, pages 175–186. Springer, 2003.

[4] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. Complex behavior at scale: an experimental study of low-power wireless sensor networks, 2002. Technical Report UCLA/CSD-TR 02-0013, UCLA.

[5] F. Kuhn, R. Wattenhofer, and A. Zollinger. Ad-hoc networks beyond unit disk graphs. In *Proc. of DIAL-M*, pages 69–78. ACM, 2003.

[6] N. Linial. Locality in distributed graph algorithms. *SIAM J. Comput.*, 21(1):193–201, 1992.

[7] D. Peleg. *Distributed computing: A Locality-Sensitive Approach*. SIAM Monographs on Discrete Mathematis and Applications, 2000.

[8] K. Sohrabi, B. Manriquez, and G. Pottie. Near ground wideband channel measurement. In *Proceedings of the EEE Vehicular Technology Conference, vol. 1*, pages 571–574, 1999.

[9] Peng-Jun Wan, Khaled M. Alzoubi, and Ophir Frieder. Distributed construction of connected dominating set in wireless ad hoc networks. In *INFOCOM*, 2002.

[10] R. Wattenhofer. Sensor networks: distributed algorithms reloaded - or revolutions? In *Proceedings of SIROCCO*, pages 24–28, 2006.

[11] A. Wiese and E. Kranakis. Local PTAS for dominating and connected dominating sets in location aware UDGs. Technical report # 07-17 at: http://www.scs.carleton.ca/ kranakis/Papers/TR-07-17.pdf.