Routing via single-source and multiple-source queries in static sensor networks

Leszek Gąsieniec Chang Su Prudence W.H. Wong Qin Xin
Department of Computer Science,
The University of Liverpool, UK
{leszek,suc,pwong,qinxin}@csc.liv.ac.uk

***Abstract*— In this paper, we introduce new geometric ad-hoc routing algorithms to route queries in static sensor networks. For *single-source-queries* routing, we utilise centralised mechanism to accomplish a query using an optimal number of transmissions $O(c)$, where c is the length of the shortest path between the source and the destination. For *multiple-source-queries* routing, the number of transmissions for each query is bounded by $O(c \log n)$, where n is the number of nodes in the network. For both single-source and multiple-source queries, the routing stage is preceded by a preprocessing stage requiring $O(nD)$ and $O(n^2 D)$ transmissions, respectively, where D is the diameter of the network. Our algorithm improves the complexity of the currently best known algorithms in terms of the number of transmissions for each query. The preprocessing is worthwhile if it is followed by frequent queries. We could also imagine that there is an extra initial power (say, batteries) available during the preprocessing stage or alternatively the position of the sensors are known in advance and the preprocessing can be done before the sensors are deployed in the field. It also worths mentioning that a lower bound of $\Omega(c^2)$ transmissions has been proved if preprocessing is not allowed [17].**

I. Introduction

Wireless ad-hoc sensor network [1, 2, 20, 23, 24] is one of the fastest growing technologies emerged in recent years. A sensor network consists of a large number of densely and arbitrarily deployed sensor nodes. The sensor nodes are self-organised and cooperate among themselves such that the sensor network is able to monitor an area of interest. Sensor networks have found applications in various areas such as environmental, medical, and military.

Sensor networks differ from traditional networks and other ad-hoc networks in many aspects. Sensor nodes usually have strong constraints like small transmission range, limited power, limited memory, and limited computational capacity. Multi-hop transmission is involved in sensor networks when two sensor nodes outside of transmission range communicate via intermediate nodes. Due to the limited memory, the topology of the sensor network is usually unknown to the sensor nodes. Moreover, there is no global identification (ID) of the sensor nodes since a large number of sensor nodes leads to too much overhead in computing the ID. Because of the power constraint, sensor networks have to minimise the number of transmissions used in communication as more transmissions result in higher power consumption.

In this paper, we study geometric ad-hoc routing [16–18] in sensor networks. The problem is to route a message from a source to a destination via some intermediate nodes. We call this communication a *query*. In geometric ad-hoc routing, it is assumed that each sensor node is informed of its own and its neighbours' coordinates [3, 5, 21]; and the source node of a query knows the position of the destination. Note that having position information in the sensor nodes (e.g., GPS [11]) becomes more and more realistic with increasing availability of inexpensive positioning systems [16]. The objective of the routing algorithm is to minimise the total number of transmissions sent for each query. The problem is non-trivial because although the coordinates of the destination node is known to the source node, the source node has no idea of what routing paths are available and which is the best path to route the query; this is because each node only has local information about its neighbours but no global information about the network topology.

A. Previous work

The simplest routing algorithm used in sensor networks is the *flooding* algorithm [4, 13] in which every node, upon receiving a message, will forward the message to all of its neighbours. The major problem of employing flooding algorithms in sensor network is that it is difficult for a sensor node to make sure that the same message will not be forwarded more than once because a sensor node cannot keep track of all messages it has received so far with its limited memory. As a result, termination of flooding cannot be controlled easily. In addition, the total number of transmissions used to route a query from the source to the destination can be huge; the lifetime of the sensor network would be much reduced by a flooding algorithm.

Another simple algorithm is the *greedy* algorithm [8, 12, 14, 22] in which a node forwards a message to the neighbour node that is the closest to the destination node. However, it has been observed that greedy algorithm does not guarantee the query can ultimately reach the destination [17].

More recent work on geometric routing tries to exploit structural property of the graph representing the network and algorithms with bounded number of transmissions are derived. These include face routing by Kranakis et al. [15], which uses $O(n)$ transmissions for a network with n nodes. Later this algorithm is enhanced to adaptive face routing [17], the number of transmissions used is bounded by $O(c^2)$, where c is the length of the shortest path between the source and the destination. However, both algorithms are not applicable in practice. Kuhn et al. [16, 18] combined face routing and greedy routing and came up with an algorithm, called GOAFR$^+$ which also uses $O(c^2)$ transmissions and this algorithm can be implemented practically. A lower bound of $\Omega(c^2)$ transmissions has been proved to be necessary to finish geometric routing [17], implying that GOAFR$^+$ is asymptotically optimal. Note that the above algorithms work on Unit-Disk Graph that possesses Gabriel Graph property (definitions will be given in Section II). Geometric routing in directed graph was also studied [6]. In this paper, we assume symmetric transmission power for the sensor nodes and we focus on undirected graphs.

B. Our results

In this paper, we adopt the Unit-Disk Graph [7] to model a sensor network [16] (formal definition to be given in Section II). We study two variants of the geometric routing problem: *single-source-queries* routing and *multiple-source-queries* routing. For the former, there is a distinguished source node and it has a number of queries to be routed to different destination nodes; for the latter, each sensor node can be a source node and it might have queries to route to different destination nodes. In either case, we know neither the shortest path C between a source s and a destination t nor its length c in advance. Note that the shortest path may be much longer than the Euclidean distance between s and t (see Figure 1).

For the single-source-queries routing, we present an algorithm whose total number of transmissions used is $O(c)$ for each query. For the multiple-source queries routing, we extend the algorithm for single-source-queries routing to a new algorithm which takes $O(c \log n)$ transmissions for each query, where n is the number of nodes in the network. For both single-source and multiple-source routing, the routing stage is preceded by preprocessing procedures requiring $O(nD)$ and $O(n^2D)$ transmissions, respectively, where D is the diameter of the network. (Note that a lower bound of $\Omega(c^2)$ transmissions has been proved if preprocessing is not allowed [17].) The preprocessing is worthwhile if it is followed by frequent queries. For example, in a network in the form of a grid with length d, the number of nodes in the network is $O(d^2)$. The preprocessing requires $O(d^3)$ and $O(d^5)$ transmissions respectively while the previous best solution takes $O(d^2)$ transmissions [16–18], thus, the preprocessing is worth-

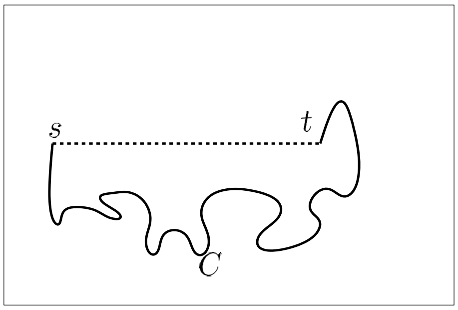

Fig. 1. The shortest path C from s to t.

while when on average there are $\Omega(d) = \Omega(\sqrt{n})$ queries per node. We could also imagine that there is an extra initial power (say, batteries) available during the preprocessing stage or alternatively the position of the sensors are known in advance and the preprocessing can be done before the sensors are deployed in the field.

C. *Organisation of the paper*

The rest of the paper is organised as follows. In section II, we recall the formal definition of the network model. Later in section III, we present a solution for the single-source query system, in which all queries originate from a distinguished central node. The solution comprises a preprocessing stage followed by an optimal (in terms of transmissions) routing stage. Finally in section IV, we show how to perform almost optimal queries originating from any node of the network.

II. MODEL

In this section, we recall the formal definition of the network model [16–18]. A sensor network is represented as a collection of n nodes arbitrarily distributed in the Euclidean plane $\mathcal{R}^2$. Precisely, a sensor network is modelled as a graph $G = (V, E)$, with the set of nodes $V \subseteq \mathcal{R}^2$ and the set of wireless undirected connections E. We assume that every node in V has the same transmission range, i.e., we adopt here the *Unit-Disk Graph model*. In this model, neighbouring nodes with edges connected are at distance at most 1. It is also assumed that the graph G possesses Gabriel Graph property [18], i.e., for any pair $(u, v) \in E$, the circle having $\overline{uv}$ as the diameter does not contain any other node of V. It is known that Gabriel Graphs are planar. As mentioned earlier, in geometric routing, every node knows its own and its neighbours' coordinates.

We consider both single-source-queries (centralised routing) as well as multiple-source-queries (distributed routing), where a query is defined as follows: A source node s wants to communicate via exchange of a control message with a destination node t, knowing only its coordinates (x_t, y_t) in $\mathcal{R}^2$. Note that s is aware of neither the topology of G nor the shortest (or in fast end) path between s and t. Furthermore, we assume that the network is static [13, 19, 23]. In this context our paper differs from the previous model [16–18], which assumes that the network is temporarily static i,.e., it does not change for the duration of each query, though between any two queries, the network topology can change arbitrarily. In sensor networks the complexity of a solution is usually expressed in terms of the total number of transmissions rather than the time used to complete a particular task. This is due to the concern of limited power of the sensor nodes.

Our model is summarised as follows:

1) Each node $v \in V$ knows its coordinates (x_v, y_v) as well as the coordinates of its neighbours.
2) The source node s knows only (x_t, y_t), the coordinates of the destination node t.
3) Each node has $O(1)$ number of neighbours.
4) Each node's memory is limited to $O(1)$ number of registers (each able to store an integer/real number) used to keep local information.
5) Nodes exchange messages in size limited to $O(1)$ integers/reals.

III. SINGLE-SOURCE-QUERIES ROUTING

In *single-source-queries* routing, a distinguished source node s wants to communicate with a destination node t, knowing only the coordinates

(x_t, y_t) of the destination node in $\mathcal{R}^2$. Note that s is neither aware of the topology of G nor the shortest path between s and t. This is due to the lack of memory in the nodes of the network.

A. The outline of our algorithm

Our algorithm consists of two phases: preprocessing phase and routing phase. In the preprocessing phase, we run the procedure called SINGLEPREPROCESS(). The objective of this procedure is to store, for any node t, a copy of its information (including a post order number of t in the BFS spanning tree rooted at the source node s) in a "special" node m such that it is easy to find a path from s to m according to the coordinates of t (precisely, (1) if the distance between s and t is c, the distance between s and m is bounded by $O(c)$; (2) according to the coordinates (x_t, y_t), the source node s could communicate with the node m in limited number of transmissions bounded by $O(c)$). In the routing phase, we run the procedure called SINGLEROUTING(). For a given destination node t, we firstly find the node m, and then communicate with the node t using an optimal number of transmissions $O(c)$ according to the additional information (a post order number of t) stored in the node m.

Note that the preprocessing procedure is done only once. After that the source node s can query any other nodes of the network using an optimal number of transmissions.

B. Data structures for handling single-source-queries

In our querying systems, we use several objects/data structures, including

- Breadth first search tree (BFS) B rooted in the source node s, spanning all nodes in G. We assume that each node $(x, y) \in G$ learns about its BFS level $bfs(x, y)$ in B during the construction of B.
- $P(x, y)$ denotes the post order number of the node (x, y) in B. This post order number is used as the physical address of the node.
- Pre-super levels
  The BFS levels in B are split into $\lfloor \log D \rfloor + 1$ pre-super levels such that each pre-super level is composed of a number of consecutive BFS levels. More formally the i-th pre-super level contains BFS levels from 2^i to $2^{i+1} - 1$, for $i = 1, ..., \lfloor \log D \rfloor - 1$. Level 0 contains BFS levels 0 and 1. Level $\lfloor \log D \rfloor$ contains BFS levels from $2^{\lfloor \log D \rfloor}$ to D. The pre-super levels are used in the construction of super levels.
- Super levels and borders
  In each pre-super level with even index, say $2i$, we choose a BFS level with the smallest number of nodes. This BFS level forms the i-th *border* . The borders spilt the BFS levels of B into $\lceil \frac{\lfloor \log D \rfloor + 1}{2} \rceil$ super levels. The i-th super level contains the BFS levels between the i-th and the $(i + 1)$-th border (but not including the $(i + 1)$-th border). Each node (x, y) in B is aware of its super level $S(x, y) \in [0, \lceil \frac{\lfloor \log D \rfloor + 1}{2} \rceil - 1]$.
- Image
  Each node (x, y) generates an image containing the key in the form of a 3-tuple $(S(x, y), x, y)$ and the content $P(x, y)$. The image is denoted by $I(x, y) = ((S(x, y), x, y), P(x, y))$.
- A priority queue PQ is a heap-like structure embedded into the BFS tree B to rearrange the images according to their keys. I.e, the key of an image stored in a parent is smaller than those stored in its children. PQ is used to sort the images within super levels in B. Using this priority queue, we rearrange the images within a particular super level i such that the following properties are satisfied.
  - Suppose there are k subtrees $T_1, T_2, T_3, ..., T_k$ rooted at the nodes in the i-th border. Give the post order number $P_{T_j}(x, y)$ to every node (x, y) in each subtree T_j, for $j = 1, ..., k$.
  - Each node $(x, y) \in T_j$ at super level i gets a rank $R_i(x, y) = (\sum_{q=1}^{j-1} |T_q|) + P_{T_j}(x, y)$.
  - Sort all images at super level i such

that images with smaller keys are placed in nodes with smaller ranks. I.e, if two images $I(x_1, y_1) = ((i, x_1, y_1), p(x_1, y_1))$ and $I(x_2, y_2) = ((i, x_2, y_2), p(x_2, y_2))$ with the lexicographic order $(x_1, y_1) < (x_2, y_2)$ [1] are moved to two locations (u_1, v_1) and (u_2, v_2) in super level i respectively, then $R_i(u_1, v_1) < R_i(u_2, v_2)$.

Lemma 1: If a node u belongs to the i-th super level, then the image of u will be stored at a node v in the i-th super level as well.

Proof: Follows directly from the sorting step above. ∎

Corollary 2: If the distance between s and t is c, then the distance between s and the node m used to store the image of t is bounded by $O(c)$.

- A search path is the longest connected path in the BFS tree B which starts from the source node. The images stored in the nodes in the i-th border will be replicated to the nodes along the search path in the i-th super level correspondingly (see Figure 2 for an example). The following lemma shows that replication requires $O(1)$ memory in each node.

Lemma 3: The nodes in the search path need $O(1)$ memory to store all the images of the nodes in the borders.

Proof: It is obvious that the size of the image in each node is of size $O(1)$. So we only need to prove that the number of nodes in the i-th border is $O(1)$ times of the number of nodes on the search path segment within the i-th super level; we denote this search path segment as P_i. We prove this by contradiction. We assume on the contrary that there are $\Omega(r)$ number of nodes in the border, where $r = 2^{2i}$. Then the number of nodes in the i-th super level is $\Omega(r^2)$. It contradicts to the result of [16], which states that there are $O(r^2)$ number of nodes in the region with radius r. Therefore the assumption is not true. The number of nodes in the i-th border must be at most $O(r)$. The number of nodes on P_i is greater than or equal to the radius r. Therefore the number of nodes in the i-th border is less than $O(1)$ times the number of nodes on P_i. ∎

[1]We say $(x_1, y_1) < (x_2, y_2)$ iff $x_1 < x_2$ or $x_1 = x_2$ and $y_1 < y_2$.

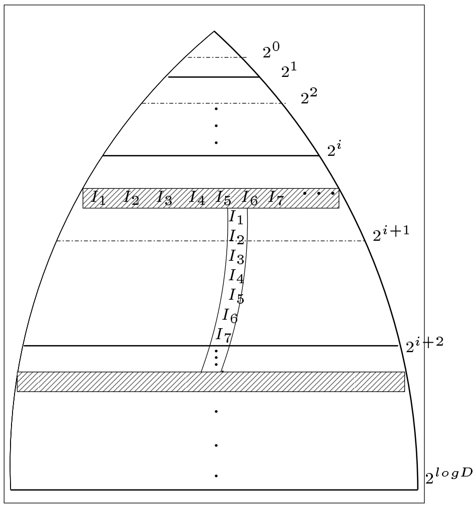

Fig. 2. A BFS tree divided into super levels and a search path in one of the super levels. (The shaded levels are the border chosen in the corresponding super levels and the images stored in the borders are replicated to the search path.)

C. Preprocessing in single-source-queries routing

In this section, we describe an $O(nD)$-transmissions preprocessing procedure for single-source-queries routing. Procedure SINGLEPREPROCESS() creates the data structures discussed in section III-B. The number of transmissions involved is bounded in Lemma 4.

Procedure SINGLEPREPROCESS(s)

1) Create BFS spanning tree B rooted at the source node s;
2) Give the post order number to each node in B;
3) Construct the search path;
4) Split the BFS levels in B into the pre-super levels;
5) Construct the borders and the super levels in B;

6) Sort the nodes in each super level, using the sorting step (described on page 5) and the priority queue PQ (described on page 4).

Lemma 4: The number of transmissions used to complete Procedure SINGLEPREPROCESS(s) can be bounded by $O(nD)$.

Proof: (Sketch) It is easy to observe that Steps 1 and 5 each takes $O(nD)$ transmissions while Steps 2 and 4 take $O(n)$ and $O(1)$ transmissions, respectively. Step 3 can be done by finding a leaf node with the largest BFS label, which takes $O(n + D)$ transmissions. The most tricky part is Step 6.

Consider the i-th super level. We first traverse all the subtrees in the BFS tree B that are in the i-th super level, such that each node learns its own rank (as defined in Section III-B); this takes $O(n)$ transmissions. Next, we build a priority queue for all the nodes in this super level, which is a heap-like structure (the degree of the node may vary but is bounded by a constant). Note that the priority queue is built in the subtree of the BFS tree from the root to the bottom level of this super level. This construction takes $O(n2^{2i})$ transmissions. With this priority queue, we can locate the nodes with increasing lexicographic order in $O(n2^{2i})$ transmissions and the corresponding images are placed in the nodes in the i-th super level with rank equals to the lexicographic order. The placement of each image takes $O(2^{2i})$ transmissions, because we can identify the appropriate subtree by looking at the search path, thus, a total of $O(n2^{2i})$ transmissions is required. Summing up the number of transmissions for all super levels, the total number of transmissions involved in Procedure SINGLEPREPROCESS() is $O(\sum_{0 \leq i \leq \lceil \log D/2 \rceil} n2^{2i}) = O(nD)$. Hence, the lemma follows. ■

D. Procedure SINGLEROUTING

After the preprocessing phase has been performed, we enter the routing phase. The following procedure describes how the routing between the source node s and any destination node t runs.

Procedure SINGLEROUTING(s, t)

1) Set $i = 0$; (starting from the first super level)
2) While t has not been found
   a) Find the first node on the search path in super level i, of which the image is greater than (x_t, y_t) (we call this image *crucial image* $I(x_c, y_c)$ recall that the images stored in the search path is in ascending order);
   b) Copy this crucial image in the query and route the query back to s;
   c) Find the node u on the border which stores $I(x_c, y_c)$;
   d) Go through the subtree routed at u; If we find $I(x_t, y_t)$, then go back to s and find t using $P(x_t, y_t)$;
   e) Set $i = i + 1$;

Theorem 5: For any destination node t, communication between the source node s and t can be completed using $O(c)$ transmissions, where c is the length of the shortest path from s to t.

Proof: By the construction of the super levels, we know the i-th super level contains at most $2^{2(i+1)}$ BFS levels. Thus, the number of super levels to be tried is at most $\lceil (\log 8c)/2 \rceil$. Together with Corollary 2, the total number of transmissions required is $O(\sum_{0 \leq i \leq \lceil (\log 8c)/2 \rceil} 2^{2i}) = O(c)$. ■

IV. MULTIPLE-SOURCE-QUERIES ROUTING

In multiple-source-queries routing, each sensor node can be a source node and it might have queries to route to different destination nodes. Unlike single-source-queries routing, there is no single distinguished source node. A simple way to exploit the algorithm for single-source queries would be to choose one central node r such that every communication between s and t is done via r. Yet there is no guarantee that the length of the route via r is comparable with the length of the shortest path between s and t. Therefore, in this section, we describe another algorithm to accomplish almost optimal multiple-source queries.

Roughly speaking, we divide the graph into clusters (which may overlap with each other); a node is chosen in each cluster as the central node and communication between the nodes in the same cluster will be handled, as in the single-source-queries case, between the nodes and the central node. To ensure that the nodes communicate within clusters efficiently (not using too many transmissions), the clusters have to be constructed in a way to preserve local distances between the nodes. For any pair of source and destination nodes s_i and t_i having a distance d_i apart, (1) they are both contained in some cluster with diameter comparable to d_i; and (2) the source node s_i can locate this cluster using a number of transmissions that is also comparable to d_i; more precisely, both quantities are $O(d_i \log n)$.

A. Preprocessing for multiple-source-queries routing

In this section, we are going to describe a preprocessing for multiple-source-queries routing. We first construct a set of clusters as mentioned above and then apply the procedure SINGLEPREPROCESS() to each of the clusters.

1) Construction of clusters: We adopt the clusters concept as used by Gąsieniec et al. [10], and Gaber and Mansour [9]. Initially we pick an arbitrary node $r \in V$ as the central node of the graph G and construct a BFS tree in G with respect to r. Recall that the BFS level of a node is j if its shortest distance to r is j. Let D be the radius of G, which is the maximum distance between r and any other node. The construction of clusters takes a parameter d, and we will run the construction for $d = 1, 2, \ldots, 2^{logD}$.

Definition 6: A *partition* $\pi(x)$ of the graph G is a division of V into *groups*, each of which comprises $4d$ consecutive BFS levels, where the first group starts from BFS level x. Precisely, for $i = 1, 2, \cdots, \lceil \frac{D-x}{4d} \rceil$, the i-th group, denoted as $G_i(x)$, contains all nodes at BFS levels j with $(i-1-x) \cdot 4d \leq j \leq (i-x) \cdot 4d - 1$.

The 2-*partition* of the graph G comprises two different partitions: $\pi(0)$ which starts with the group $G_1(0)$, and $\pi(2d)$ which starts with the group $G_1(2d)$.

Note that BFS levels $0, 1, \ldots, x-1$ are excluded from the partition $\pi(x)$. For any group $G_i(x)$, its *top* level is $(i-1-x) \cdot 4d$, and its *bottom* level is $(i-x) \cdot 4d - 1$. Note that $G_i(x)$ is not necessarily connected. In each group $G_i(x)$, we first construct some pre-clusters, based on which we construct the clusters.

Definition 7: For each node u belonging to the top level of $G_i(x)$, the *pre-cluster* S_u^i is defined to be the set of all nodes in $G_i(x)$ whose distance from u is at most $4d$.

Now we briefly describe the idea of Gąsieniec et al. [10] of how to construct the *clusters* by growing appropriate pre-clusters. The growing algorithm executes in $O(\log n)$ stages. In Stage j, where $1 \leq j \leq \log n$, a collection of clusters C_*^j would be created. An arbitrary pre-cluster is chosen as a core of a new cluster C_0^j. The core C_0^j is extended, by adding a layer all pre-clusters that intersect with C_0^j or are at distance at most 1 from C_0^j, to form a new core and is then further extended similarly. The extension continues as long as the number of new nodes to be added is at least the number of nodes already present in the core C_0^j; otherwise, the extension of C_0^j is terminated and the pre-clusters in the new layer are promoted for consideration in Stage $j+1$. We then grow the clusters $C_1^j, C_2^j, \ldots$ similarly until all pre-clusters are either included in a cluster or promoted to Stage $j+1$.

It can be easily observed that each cluster is a union of some pre-clusters; each pre-cluster belongs to exactly one cluster; and each cluster is a connected sub-graph of G. A more detailed analysis of the growing process gives the following lemma.

Lemma 8: [10] For every d considered, (a) the diameter of each cluster is $O(d \log n)$, (b) every node only belongs to $O(\log n)$ number of clusters, and (c) for any two nodes whose shortest path between them is of length d, then in at least one of the partitions of the 2-*partition*, there exists at least one cluster that contains both nodes.

Lemma 9: The number of transmissions required to construct all clusters for different values of d is $O(n^2 D)$.

Proof: (Sketch) The crucial step in the construction is the growing of a core cluster. First consider the growing for a particular d value. In Stage 1, to grow a core cluster, we start traversal from the (newly added) nodes in the core cluster for distance $4d$ to reach all other pre-clusters that have intersection or with distance one apart. This can be done by using $O(n'nd)$ transmissions, where n' is the number of (newly added) nodes in the core cluster. Notice that the clusters constructed in the same stage are all disjoint. Therefore, Stage 1 requires at most $O(n^2 d)$ transmissions. After Stage 1, there are at most $n/2$ nodes remained. Therefore, the number of transmissions required for a particular d value is $O(\sum_{0 \le j \le \log n}((n/2^j)^2 d)) = O(n^2 d)$. Hence, the total number of transmissions required is $O(\sum_{0 \le i \le \log D}(n^2 2^i)) = O(n^2 D)$. ■

2) Applying SINGLEPREPROCESS*() in the clusters:* After constructing the clusters, we apply Procedure SINGLEPREPROCESS() to each of these clusters. This is done by arbitrarily picking one node r in each cluster, say the central node of the first pre-cluster chosen in the corresponding cluster; and then applying SINGLEPREPROCESS(r). The following lemmas state the number of transmissions required for this preprocessing and the memory requirement needed in each node.

Lemma 10: Applying SINGLEPREPROCESS() to all clusters requires $O(nD \log^2 n)$ transmissions.

Proof: By Lemma 4, applying SINGLEPREPROCESS() on a cluster of size n' and diameter $d \log n$ takes $O(n'd \log n)$ transmissions. By Lemma 8 (b), every node belongs to at most $O(\log n)$ clusters. Therefore, the total number of transmissions required for applying SINGLEPREPROCESS() for a particular d is $O(nd \log^2 n)$. Counting all d values we use, the total number of transmissions for applying SINGLEPREPROCESS() is $O(nD \log^2 n)$; thus, the lemma follows. ■

Lemma 11: For any node u in the graph, the memory size required to store the information of all clusters that u belongs to is $O(\log n \log D)$.

Proof: By Lemma 8 (b) and the fact that there are $O(\log D)$ different d values, a node only belongs to $O(\log n \log D)$ clusters. Together with Lemma 3 which states that $O(1)$ memory is required for one cluster, the total memory size required is $O(\log n \log D)$; thus, the lemma follows. ■

Remark 12: The multiple-source-queries can be performed in constant space with a poly-logarithmic transmission overhead.

The memory consumption in each node of the network can be reduced to a constant in the following way. Note, that the $O(\log n \log D)$ space requirement comes from the need of use of the clustering system, where each node has to remember the clusters to which it does belong to. In the space efficient solution the set of single nodes is replaced by the set of *super-nodes*, where each super-node is a small cluster of neighbouring nodes of size $\Theta(\log n \log D)$. The set of super-nodes is computed as follows. We first build a spanning tree T_S in $G = (V, E)$. Then the algorithm partitions the set of nodes into super-nodes by cutting branches of T_S of size $O(\log n \log D)$. This is always feasible since the maximum degree of T_S is $O(1)$. The super-nodes (each based on a branch from T_S) form the set of nodes $\bar{V}$ in the new graph $\bar{G} = (\bar{V}, \bar{E})$. In $\bar{G}$, for any $v, w \in \bar{V}$ a pair $(v, w) \in \bar{E}$ iff there exist $x, y \in V$, s.t., $x \in v, y \in w$ and $(x, y) \in E$. Note that the maximum degree in $\bar{G}$ is bounded by $O(\log n \log D)$. The original nodes in each super-node play a role of single cells in a distributed memory. The new graph $\bar{G}$ can be preprocessed similarly as G. Now, the information about the $O(\log n \log D)$ clusters can be distributed evenly within each super-node, s.t., each original node stores information about a constant number of the clusters. However on the downside there will be a poly-logarithmic transmission overhead related to the larger degree and to the cost of internal search for an appropriate information in the distributed memory of each super-node.

B. *Procedure* MULTIROUTING

After the construction of clusters and the preprocessing, we enter the routing stage. Procedure MULTIROUTING() describes how the routing between any pair of source s and destination t runs.

Procedure MULTIROUTING(s, t)

1) Set $d = 1$;
2) While the cluster containing both s and t has not been found
   a) For every cluster with diameter $O(d \log n)$ that s belongs to
      i) Based on the BFS tree in this cluster, s sends a message to the root r with information including the coordinates of the destination t;
      ii) Apply SINGLEROUTING(r, t);
      iii) If t can be found, the routing is completed and the procedure terminates; otherwise, continue with the next cluster;
   b) Set $d = 2 * d$;

The following theorem gives the performance of Procedure MULTIROUTING().

Theorem 13: For any pair of source s and destination t, the communication between s and t can be completed using $O(c \log n)$ transmissions, where c is the length of the shortest path from s to t.

Proof: Procedure MULTIROUTING starts from $d = 1$ and doubles d in each iteration. By Lemma 8 (c), the largest value of d tried by MULTIROUTING is at most $2c$. Together with Theorem 5, the total number of transmissions required is $O(\sum_{0 \leq i \leq \lceil \log 2c \rceil} 2^i \log n) = O(c \log n)$. Thus, the theorem follows. ■

REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cyirci. A survey on sensor networks. *IEEE Communications Magazine*, pages 1–13, August 2002.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cyirci. Wireless sensor networks: A survey. *Computer Networks*, pages 393–422, 2002.

[3] P. Bahl and V. N. Padmanabhan. RADAR: An in-building RF based user location and tracking system. In *Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 775–784, 2000.

[4] D. Braginsky and D. Estrin. Rumor routing algorithm for sensor networks. In *Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications*, pages 22–31, Atlanta, Georgia, USA, September 2002.

[5] N. Bulusu, J. Heidemann, and D. Estrin. GPS-less low cost outdoor localization for very small devices. *IEEE Personal Communications*, 7(5):28–34, 2000.

[6] E. Chávez, S. Dobrev, E. Kranakis, J. Opatrny, L. Stacho, and J. Urrutia. Route discovery with constant memeory in oriented planar geometric networks. In *Proceedings of the First International Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS)*, pages 147–156, Turku, Finland, July 2004.

[7] B.N. Clark, C.J. Colbourn, and D.S. Johnson. Unit disk graph. *Discrete Mathematics*, 86:165–177, 1990.

[8] G. Finn. Routing and addressing problems in large metropolitan-scale internetworks. Technical Report ISI/RR-87-180, USC/ISI, March 1987.

[9] I. Gaber and Y. Mansour. Centralized broadcast in multi-hop radio networks. *Journal of Algorithms*, 46(1):1–20, 2003.

[10] L. Gąsieniec, E. Kranakis, A. Pelc, and Q. Xin. Deterministic M2M multicast in radio networks. In *Proceedings of the Thirty-First International Colloquium on Automata, Languages and Programming*, pages 670–682, Turku, Finland, 2004.

[11] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins. *Global Positioning System: Theory and Practice*. Springer, 5th edition, 2001.

[12] T. Hou and V. Li. Transmission range control in multihop packet radio networks. *IEEE Transactions on Communications*, 34(1):38–44, 1986.

[13] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the Sixth International Conference on Mobile Computing and Networks*, pages 56–67, Boston, Masschusetts, August 2000.

[14] B. Karp and H.T. Kung. GPRS: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the Sixth International Conference on Mobile Computing and Networks*, pages 243–254, Boston, Masschusetts, August 2000.

[15] E. Kranakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. In *Proceedings of the Eleventh Canadian Conference on Computational Geometry*, pages 51–54, August 1999.

[16] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger. Geometric ad-hoc routing: Of theory and practice. In *Proceedings of the Twenty-Second ACM Symposium on the Principles of Distributed Computing*, pages 63–72, Boston, Massachusetts, July 2003.

[17] F. Kuhn, R. Wattenhofer, and A. Zollinger. Asymptotically optimal geometric mobile ad-hoc routing. In *Proceedings of the Sixth International Workshop on Discrete Algorithm and Methods for Mobility*, pages 24–33, Atlanta, Georgia, USA, September 2002.

[18] F. Kuhn, R. Wattenhofer, and A. Zollinger. Worst-case optimal and average-case efficient geometric ad-hoc routing. In *Proceedings of the Fourth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 267–278, Annapolis, Maryland, June 2003.

[19] D. Liu and P. Ning. Location-based pairwise key establishments for static sensor networks. In *Proceedings of the Sixth ACM Workshop on Security of Ad hoc and Sensor Networks*, pages 72–82, Fairfax, Virginia, 2003.

[20] G.J. Pottie and W.J. Kaiser. Wireless integrated sensor networks. *Communications of ACM*, 43(5):51–58, May 2000.

[21] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *Proceedings of the Sixth International Conference on Mobile Computing and Networks*, pages 32–43, Boston, Masschusetts, August 2000.

[22] H. Takagi and L. Kleinrock. Optimal transmission ranges for randomly distributed packet radio terminals. *IEEE Transactions on Communications*, 32(3):246–257, 1984.

[23] S. Tilak, N.B. Abu-Ghazaleh, and W. Heinzelman. A taxonomy of wireless micro-sensor network models. *ACM Mobile Computing and Communications Review*, 6(2):28–36, April 2002.

[24] M. Tubaishat and S. Madria. Wireless sensor networks: A survey. *Potentials IEEE*, 22:20–23, April 2003.