Exact Distributed Voronoi Cell Computation in Sensor Networks *

Boulat A. Bash and Peter J. Desnoyers
Department of Computer Science
University of Massachusetts – Amherst
Amherst, Massachusetts 01003–9264
{boulat,pjd}@cs.umass.edu

ABSTRACT

Distributed computation of Voronoi cells in sensor networks, i.e. computing the locus of points in a sensor field closest to a given sensor, is a key building block that supports a number of applications in both the data and control planes. For example, knowledge of Voronoi cells facilitates efficient methods for computing the piece-wise approximation of a field, whereby each sensor acts as a representative for the set of points in its Voronoi cell; awareness of Voronoi boundaries and Voronoi neighbors is also useful in load balancing and energy conservation. The methods currently advocated for distributed Voronoi computation in sensor networks are heuristic approximations that can introduce significant inaccuracies that are difficult to rigorously quantify; we demonstrate that these methods may err by a factor of 5 or more in some circumstances. We present and prove an exact method which eliminates these inaccuracies, at the cost of increased messaging overhead, but without necessitating contact with the entire network. To our knowledge, this is the first distributed algorithm that computes accurate Voronoi cells without requiring all-to-all communication. We implement it as a TinyOS module and quantitatively analyze its performance.

Categories and Subject Descriptors

E.1 [**Data Structures**]: Distributed data structures—*Voronoi diagrams*; H.4.3 [**Information Systems Applications**]: Communications Applications

General Terms

Algorithms, Design, Performance

Keywords

Sensor networks, Voronoi diagrams, In-network processing and aggregation

*This work was supported in part by the Engineering Research Centers Program of the National Science Foundation under NSF Cooperative Agreement No. EEC-0313747, and NSF grants ANI-0085848 and EIA-0080119.

IPSN'07, April 25-27, 2007, Cambridge, Massachusetts, USA.

1. INTRODUCTION

The distributed and resource-constrained nature of sensor networks often demands unique solutions to otherwise well-understood problems. One such problem is the accurate determination of Voronoi cells. In the context of sensor networks, a sensor's Voronoi cell is the area closer to that sensor than to any other sensor. This area has many natural interpretations in sensor networks; for example, it might be the area where that sensor has the most authoritative or representative readings. However, while the computation of Voronoi cells is a standard problem in computational geometry, it is much less understood in the distributed setting of sensor networks. Indeed, prior to this work the only distributed algorithm for the exact computation of a Voronoi cell necessitated contact with all of the sensors in the network [4]. Several recently proposed heuristic approaches require fewer messages [25, 28], but provide no guarantees of the correctness of their result. In this paper we present an algorithm which is efficient in typical cases, but is still provably exact.

A sensor network typically is comprised of a large number of small battery-powered systems or *motes* [15] distributed over a wide area, communicating over low-powered, short-range radios. These motes are often aware of their own locations, either through GPS or through an array of inference techniques [21, 22]. They can perform a variety of measurements of their environment, and their collective capability can be quite substantial when there are many nodes in the network. Indeed, proponents of sensor networks have proposed networks of thousands or hundreds of thousands of sensors communicating and working together. However the extreme scale involved, combined with the low bandwidth between sensors and node/link failures, result in many challenges. In particular, protocols for sensor networks must be able to tolerate failures while being quite parsimonious in their communication.

Sensor network applications typically wish to learn about the environment being sensed, not the sensors or systems themselves, whether collecting individual measurements or computing averages or other functions. For example, consider a query for the average temperature over a region where sensors are distributed in a non-uniform pattern. As pointed out in [11], the average over an area is not the same as the average over the sensors, and in practice, these two quantities can be quite different. A simple average of all the sensor readings, for example, will be biased towards areas with more sensors.

Instead, a more accurate approximation to the average

temperature over the region could be reached by weighting each sensor's reading proportional to the area which is closest to that sensor. These regions averaged over are in fact the Voronoi cells of each individual sensor. Thus Voronoi cells give a natural weighting scheme for such applications.

Many other problems could benefit from improved Voronoi cell algorithms, as well. There have been various efforts to bridge over techniques from spatial databases [12] to sensor networks; problems which are simplified given the Voronoi cells of the network include range, nearest neighbor and reverse nearest neighbor queries [9, 26], and spatial aggregation [13, 25]. Accurate Voronoi cells aid various monitoring problems, including target localization and tracking [5] and random sampling of sensors [2], as well as problems of the control plane such as energy conservation [28] and load balancing [3]. In addition, performance of geographic routing protocols such as GPSR [16] may be improved given knowledge of Voronoi cells.

For each of the problems enumerated above, the correctness of the results hinges on the correct computation of the Voronoi cells. But given the communication and computational constraints of sensor networks, the well-understood centralized and even parallel algorithms for computing the set of Voronoi cells (a *Voronoi diagram*) are inapplicable. Instead, the literature so far has either relied on lightweight approximations, such as only considering the set of nodes within k hops, for some small k [25, 28], or exact but expensive methods which involve contacting the entire network, at a prohibitive cost [4]. In this paper, we present a distributed algorithm that computes provably exact Voronoi cells without necessarily contacting the entire network; to our knowledge, this is the first such method in its class. The algorithm begins with an initial approximation of the local Voronoi cell at each node, based on neighboring nodes. Whereas existing heuristic approximations terminate at this point, our algorithm instead leverages geographic routing primitives such as GPSR [16] to systematically refine the Voronoi cell and verify its correctness.

The remainder of this paper is organized as follows: in Section 2 we formally define Voronoi cell and discuss its construction. We discuss the existing distributed solutions in Section 3 and present our algorithm in Section 4. Practical implementation issues and experimental results are presented in Section 5, and Section 6 concludes.

2. VORONOI CELLS AND DIAGRAMS

We begin our discussion with basic definitions, adapted from [8].

Definition 1. (Voronoi Cell) The *Voronoi cell* of a node p with respect to a set of nodes N, denoted $V_N(p)$, is the set of points in the plane which are closer to p than any node in $N - \{p\}$.

Note that $V_N(p)$ is not necessarily closed.

Fact 1. For any node $p \in N$, the Voronoi cell $V_N(p)$ is unbounded if and only if p is on the convex hull of N. If $V_N(p)$ is bounded, then $V_N(p)$ is a convex polygon.

For sensor network applications, we are typically interested in a sensor field A of bounded extent, where $N \subseteq A$. In this case, we wish to calculate the region $A \cap V_N(p)$ which is always bounded.

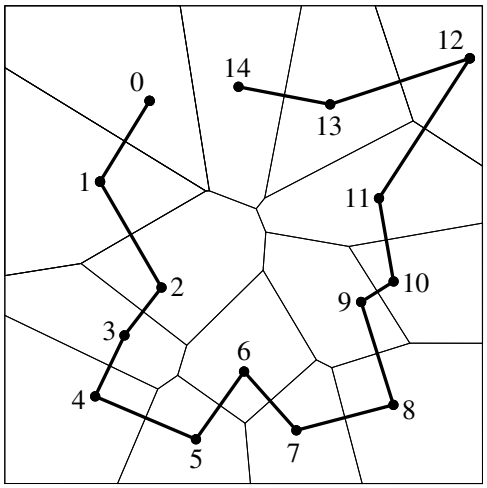


Figure 1: Voronoi diagram of a 15-node topology where two nodes sharing a Voronoi cell edge (Delauney neighbors) are 14 hops apart.

Definition 2. (Voronoi vertex) A Voronoi vertex is a vertex of the convex polygon that is the boundary of a given Voronoi cell. The set of Voronoi vertices for node p defines its Voronoi cell.

Definition 3. Let $H(p_1, p_2)$ be the half-plane defined by the perpendicular bisector of $p_1 \leftrightarrow p_2$ containing p_1.

Fact 2. A point x is closer to p_1 than p_2 if and only if x lies in $H(p_1, p_2)$.

THEOREM 1 (VORONOI CELL CONSTRUCTION). *For any node $p \in N$,*

$$V_N(p) = \bigcap_{p' \in (N-p)} H(p, p').$$

Property 1. (Voronoi Cell Monotonicity)
For any sets of nodes N_1, N_2 in which $N_1 \subseteq N_2$,

$$\forall p \in N_1, V_{N_2}(p) \subseteq V_{N_1}(p).$$

That is, the Voronoi cell of node p in Definition 1 decreases monotonically as more nodes are added to set N. If only a subset of the nodes in the network are known, then an upper bound is computed, in the sense that the area returned contains the true Voronoi cell.

Definition 4. (Voronoi Diagram) The Voronoi diagram D_N of set N is the set of Voronoi cells of nodes in N.

Fig. 1 shows the Voronoi diagram of a 15-node sensor network. The sum of areas of exact Voronoi cells will yield the area of the network field A. If some cells are not computed exactly, due to knowledge of only a subset of nodes in N, then, by Property 1, they will overlap.

Fact 3. (Trivial Voronoi Diagram exactness)
If $D_N = \{V_N(p_i) | p_i \in N\}$, then $D_N = A$.

Fact 3 states that if the Voronoi cell of each node is constructed with respect to all other nodes in the network N,

then each cell will be exact. Thus, the set of Voronoi cells will be a partition of the sensor field A. Therefore, fact 3 provides a method to compute Voronoi cells. It is the basis of the existing heavyweight approaches for distributed Voronoi computation. However, each node needs only a subset of nodes to compute its cell, namely the set of nodes with which it shares Voronoi cell edges (also known as *Delaunay neighbors* in computational geometry). We call this subset D_i for node p_i and we can now state our final fact:

Fact 4. (Voronoi cell exactness) If $V_X(p_i)$ and $X = D_i$, then $V_X(p_i)$ is exact, and if $D_N = \{V_{D_i}(p_i)|p_i \in N\}$, then $D_N = A$.

Note that D_i may not be a subset of communication neighbors. For example, this is the case for node 0 on Fig. 1. Our distributed algorithm performs a search for nodes in D_i using geographic routing.

3. EXISTING APPROACHES

Current distributed algorithms for Voronoi cell computation [4, 14, 25] use variations of the following basic k-hop neighbor heuristic.

Algorithm 1 k-hop neighbor Voronoi

1: Request locations of all neighbors within k hops of the base station at point p.
2: Let K be the set of points returned by Step 1.
3: Return $V_K(p)$.

By Property 1, it follows that the Voronoi cell supported by members of the k-hop neighborhood encloses the actual cell $V(p)$. However, it is straightforward to construct example graphs for which all Voronoi neighbors of p are not contained within the k-hop neighborhood, for any $k < n-1$ (see Fig. 1). Also, as our experimental results in Section 5.2 indicate, use of this heuristic with the moderate values of k recommended in practice often leads to approximate Voronoi cells where there is significant overlap.

In [14, 25] the node uses its 1-hop neighbors to construct initial Voronoi cell and refines it gradually as it captures packets containing location information from other nodes in the network. The authors do not claim that their method outputs the exact Voronoi cell. In [4], the authors mitigate this problem by having every node flood the network with d beacons containing location information, where d is the network diameter in hops. Since every node knows of every other node, their algorithm is exact, but at the transmission cost of $O(d \times n^2)$ messages.

4. DISTRIBUTED VORONOI COMPUTATION

We now present a framework and algorithms for exactly constructing Voronoi cells. This framework generalizes earlier "collection" based efforts to construct Voronoi diagrams in a distributed fashion, and makes explicit the requirements for their success. We then leverage the geographic routing methods of [16] to build an exact method.

Our key idea is to use the monotonicity property of Voronoi cells, stated as Property 1. For each sensor p, we initialize its tentative Voronoi cell to the region induced by a small subset of nodes. By monotonicity, this region is known to enclose the actual Voronoi cell $V(p)$. Then, while an additional node exists whose inclusion reduces the extent of the tentative Voronoi cell of p, we add that node to the subset. The procedure terminates when no such node exists, in which case $V(p)$ has been identified. This algorithm is presented below as Algorithm 2, and relies on an appropriate initialization method, and an efficient method for identifying a node that improves the Voronoi cell of p. We present those next.

Algorithm 2 Exact Voronoi

1: Let S_0 be an initial set of nodes that yields a bounded Voronoi cell $V_{S_0}(p)$.
2: Invoke node discovery method D to identify a set of nodes ΔS which reduce $V_{S_0}(p)$.
3: **if** $\Delta S = \emptyset$ **then**
4: Return $V_{S_0}(p)$.
5: **else**
6: Set $S_0 = S_0 \cap \Delta S$ and repeat Step 2.
7: **end if**
8: Return $V_{S_0}(p)$.

4.1 Initialization

We assume that each sensor is aware of the boundaries of network field A, which is considered to be the Voronoi cell at time zero. Since we assume that our network contains more than one node and is connected, each node has at least one communication neighbor which can conveniently reduce the time-zero cell. In our experiments, A reduced by all 1-hop communication neighbors is used as the initial cell. In theory, any convex polygon that contains $V(p)$ can be used to initialize this algorithm.

4.2 Incremental Update Procedure

Definition 5. Consider the Voronoi cell $V_X(p)$ of node p induced by a subset of nodes X. We say that addition of node $q \notin X$ *reduces* p's Voronoi cell iff $V_{X\cup\{q\}}(p) \subset V_X(p)$, that is, perpendicular bisector of $p \leftrightarrow q$ defining $H(p,q)$ intersects $V_X(p)$.

LEMMA 1. *Consider a bounded Voronoi cell $V_X(p)$. A node q reduces $V_X(p)$ iff there exists a vertex z of $V_X(p)$ such that $d(q,z) < d(z,p)$.*

PROOF. We prove the only-if direction first. Assume there exists a vertex z of $V_X(p)$ such that $d(q,z) < d(z,p)$. Consider the Voronoi cell after addition of q: $V_{X\bigcup\{q\}}(p)$. Point z lies strictly closer to q than to p, therefore it lies outside of p's Voronoi cell, and hence node q reduces p's Voronoi cell.

We now prove the if direction. Assume there exists a node q that reduces $V_X(p)$. Now assume for the sake of contradiction that there does not exist a vertex z of $V_X(p)$ such that $d(q,z) < d(z,p)$. Then $d(q,z) \geq d(z,p)$ for all vertices $z \in V_X(p)$. However, q can not reduce $V_X(p)$ in this case, since perpendicular bisector defining $H(p,q)$ can not intersect $V_X(p)$, thus contradicting the original assumption. □

Lemma 1 gives necessary and sufficient conditions for a procedure which determines if a node exists that reduces p's Voronoi cell. We embody those conditions in the following class of methods that locate such a node if one exists.

Definition 6. (Admissible Discovery Method) Consider the Voronoi cell $V_X(p)$ of a node induced by a subset of nodes X. A node discovery method D is admissible if it always returns a node $p' \notin X$ that reduces $V_X(p)$ if such a node exists, and nothing otherwise.

It is quite easy to realize an admissible discovery method given a geographic routing protocol such as GPSR [16].

Fact 5. (Geographic Routing [16, 23]) A GPSR packet sent to an arbitrary point x in the plane is always routed to the reachable node p minimizing $d(p, x)$, where d denotes Euclidean distance.

Algorithm 3 Discovery by Vertex Verification

1: **for** each vertex v of $V_X(p)$ **do**
2: Use GPSR to identify the node p' closest to v.
3: **if** $d(p', v) < d(p, v)$ **then**
4: Report p' as a node that reduces p's Voronoi cell.
5: **end if**
6: **end for**
7: Report that no vertices reduce p's Voronoi cell.

From Fact 5, this is an admissible discovery method. It clearly terminates after $|S|$ calls, since it eliminates one node from consideration as a Voronoi neighbor at each call. Also, in practice, one would not recheck any Voronoi vertex for which a nearer neighbor was not identified.

Discovery by vertex verification works and is conceptually simple, but can be expensive. If the initial polygon is large, such as when using the cells induced by 1-hop neighbors in a sparsely connected network, initial probes must cross the network diameter. Furthermore, this discovery method can result in a bottleneck if all sensors initially start with similar Voronoi neighbors and probes are sent to the same small sets of nodes. The following improvement to Algorithm 3 heuristically addresses both of these issues while remaining an admissible strategy. This is one of the variants that we implement in our experimental work.

Algorithm 4 Discovery by Vertex Verification

1: **for** each vertex v of $V_X(p)$ **do**
2: Use GPSR to route towards point v.
3: As soon as the GPSR packet is routed through any node q such that $d(q, v) < d(p, v)$, return q as a node that reduces p's Voronoi cell, and stop the probe.
4: **end for**
5: Report that no vertices reduce p's Voronoi cell.

Since in a non-degenerate Voronoi diagram[1] most Voronoi vertices are present in the cells of three nodes, Algorithm 4 usually sends three probes to verify a single vertex. The bandwidth usage can be reduced if nodes with adjacent cells cooperate. The following algorithm requires additional state per node in the order of the size of the Voronoi cell.

Algorithm 5 Cooperative Vertex Verification

1: **for** each vertex v of $V_X(p)$ **do**
2: Notify nodes n and m where $v \in V(n)$ and $v \in V(m)$ of intent to verify v.
3: Invoke Steps 4 and 4 of Algorithm 4 to verify v.
4: Report output of Step 5 to n and m, including q if it reduces $V_X(p)$
5: **end for**
6: Report that no vertices reduce p's Voronoi cell.

In addition to Algorithm 4, we implement Algorithm 5, and compare the two algorithms head-to-head. In our implementation of Algorithm 5, nodes n and m use a timeout mechanism to recover in cases where the verification report from node p is not heard.

5. EXPERIMENTAL EVALUATION

We now present the experiments involving our algorithm. We start by outlining the methodology behind our simulations in Section 5.1, and then examine Voronoi cell overestimates of the baseline k-hop Algorithm 1 in Section 5.2. We compare the non-cooperative and cooperative versions of our algorithm in Section 5.3 and conclude the section with the analysis of TOSSIM experiments in Section 5.4.

5.1 Methodology

Our experiments were performed using the SENS [20] simulation platform, developed at Boston University, and the TinyOS Simulator (TOSSIM) [18].

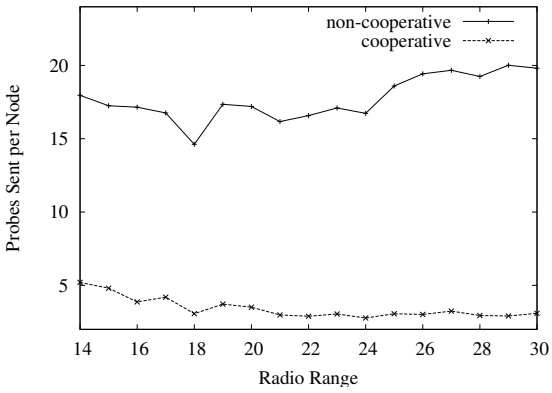
The experimental setup for SENS simulations consists of 100 nodes placed uniformly at random on a 100×100-unit field, communicating via fixed-range idealized radios. We generate 17 different random topologies, each for a different radio range $r = \{14..30\}$. Finally, 50 5-unit length radio-opaque obstacles are placed on the field, with centers chosen uniformly at random and equiprobably oriented N/S or E/W. This procedure, which we borrow from [17], adds a degree of realism to the synthetic topologies, since the idealized radio model is known to be a poor approximation of wireless connectivity [1]. We then use CLDP [17] to extract a planar subgraph of the resulting network topology for use by GPSR [16].

TOSSIM allows actual TinyOS implementations to be used in simulation, and our TOSSIM experiments used an implementation of Algorithm 5 along with CLDP and GPSR modules from the USC Embedded Networks Laboratory [27].

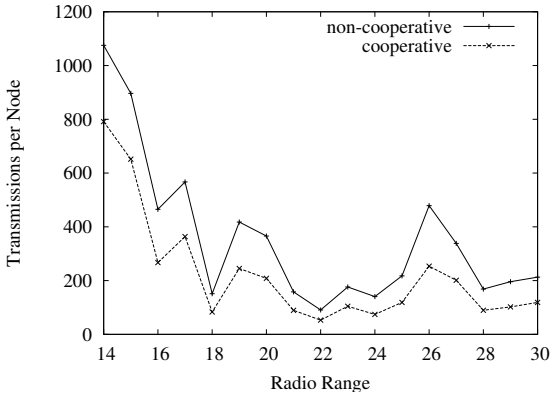
Topologies used in our TOSSIM simulations consist of 20 nodes each, with nodes placed uniformly at random on a 60×60-unit field, connected by idealized radios of ranges between 15 and 40 units in increments of 5. Again, some links were disconnected by 10 4-unit obstacles, randomly placed in the same manner as in the SENS experiments.

Five topologies were generated at each radio range, for a total of 30. Simulations were run 20 times on each of the 30 topologies, at bit error rates of 0, 0.037%, and 0.077%, corresponding to packet loss rates for 36-byte packets of 0, 10, and 20 percent. The entire number of simulation runs was $30 \times 20 \times 3$ or 1,800 runs.

[1]The condition for non-degeneracy is that no point in the plane is equidistant to more than three nodes. This condition would be violated, for instance, if one could draw a square by connecting four nodes with line segments. In that case, the center of the square would be a vertex present in Voronoi cells of these four nodes. In this work, we assume non-degeneracy holds; in reality, it can be easily ensured via small random perturbations to node locations.



(a) Average number of probes sent by each node



(b) Average number of messages sent by each node

Figure 3: Messaging efficiency comparison between non-cooperative and cooperative Voronoi cell construction algorithms (Algorithms 4 and 5, respectively). The same topologies for each radio range were used by both algorithms. The results were computed by summing the number of messages sent by each node and dividing by the number of nodes. Values represent averages of 10 runs per topology.

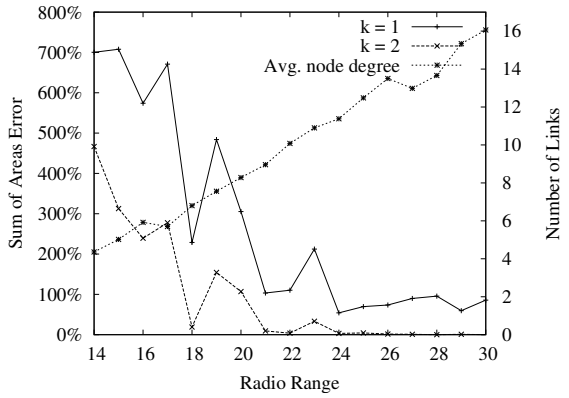


Figure 2: Relationship between the overestimate of the sum of areas of Voronoi cells and network connectivity. Neither 1-hop nor 2-hop neighbors provide enough information to obtain accurate Voronoi cells in sparse topologies, and computation using 1-hop neighbors under-performs the use of 2-hop neighbors on all tested topologies.

5.2 Validation of Motivation

We first examine the cell-overestimate errors of Algorithm 1. We use values of $k = \{1, 2\}$, since $k = 1$ is cited in previous work on distributed voronoi cell computation [25, 14], and $k = 2$ is reasonable given that nodes in several variants of link-state routing algorithms use two-hop neighborhood information [6, 24]. Our evaluation metric is the error in the network field area if summed Voronoi cell areas are used to compute it, defined as follows:

$$E_N^k = \frac{\sum_{i \in N} A(V_i^k)}{100 \times 100} \times 100\% \quad (1)$$

where $A(V_i^k)$ is the area of i-th node's k-hop Voronoi cell. Fig. 2 shows the relationship between the overestimate of our metric and connectivity of the 17 random 100-node SENS topologies described in the previous section. As expected, Algorithm 1 performs poorly in sparse settings for both values of k. Notably, even when $k = 2$, some Voronoi cells are significantly overestimated in moderately dense topologies. Algorithm 1 returns dismal results across all topologies for $k = 1$. Since sensor networks are generally sparse, if one wants to perform operations requiring accurate Voronoi cells (such as a piece-wise approximation of a field [13]) one can not rely on k-hop Voronoi computation algorithm. Thus, there is a strong motivation to systematically look beyond close neighbors while constructing Voronoi cells.

5.3 Validation of the Value of Cooperation

Having established the necessity for exact Voronoi cell computation in some applications, we proceed to evaluate two different implementations of our solution. Section 4 describes two algorithms: Algorithm 4 which sends a probe (a GPSR packet) towards each unverified vertex, and Algorithm 5 which notifies its Voronoi neighbors with whom it shares a vertex before sending a probe towards it, and notifies them again with the result. Algorithm 4 is conceptually simpler since it requires no cooperation among nodes. However, Algorithm 5 has the potential to substantially reduce messaging by eliminating the redundant probes.

The results of SENS evaluation on Fig. 3 confirm the latter assertion. Without cooperation, each vertex v must be verified once by each node p_i which contains v in its approximate Voronoi cell at some step of Voronoi cell construction. With cooperation, only one node (say p_0) will verify v, and in the process send two messages to each p_i which contain v in their approximate Voronoi cells. Since each probe verifying v must at minimum visit each node whose true Voronoi cell contains v, in the typical case the cooperative algorithm will require fewer—sometimes many fewer—transmissions.

The decrease in the number of probes sent when the cooperative approach is used is shown in Fig. 3(a); the effect on the total messaging rate (which is different due to multihop transmission of probes) is shown in Fig. 3(b). For our TOSSIM implementation the cooperative approach of Algorithm 5 was chosen, due to its superior performance.

5.4 TOSSIM Simulations

We will now describe the implementation of the Voronoi module for TinyOS, the results from experiments, and the limitations of this system.

5.4.1 Implementation

We implemented Algorithm 5 as a module for TinyOS 1.1. Our module runs on top of the USC implementation of GPSR and CLDP protocols for TinyOS [27], and provides interfaces for an application to send GPSR messages as well as to access the Voronoi cell. We use the tightly-coupled routing mode in USC GPSR which introduces considerable delay during message transmission. However, this seems to be the only non-invasive method of stopping the messages (and performing the comparison in Step 4 of Algorithm 4) in the higher layer.

Initially, unverified vertices are placed on a verification queue in random order. As new Voronoi neighbors are found via probing, new vertices are placed at the end of this queue. Vertices being probed are taken off the verification queue and a timing mechanism is associated with each probe: if three consecutive probes for the same vertex do not return, the vertex is returned to the end of queue. If a message is received from another node stating the intent to start verifying a common vertex, this vertex is also taken off the verification queue and associated with a countdown timer equal to slightly more than three probe time-out periods. Upon expiration of this timer, the vertex is placed back on the queue. This mechanism ensures that each node will eventually verify its cell, providing robustness against packet losses—though it is possible that two nodes verify the same shared vertex. The complete implementation is slightly over 2,500 lines of nesC code.

5.4.2 Results

The results of TOSSIM simulation show that Algorithm 5 performs better on average in sparse network setting than the only other existing exact Voronoi computation approach, which involves flooding the network d times, as described in Section 3 and in [4]. We present the results for two metrics: the number of probes and the total number of packets sent by the Voronoi module on each node. Each probe corresponds to steps 4 and 4 in Algorithm 4; at each hop GPSR passes the probe through the Voronoi module, resulting in a packet transmission at each node in the path. Thus the "number of transmissions" metric, when summed over all nodes, gives the total number of radio packets sent during algorithm execution, exclusive of error retransmission, fragmentation, and routing protocol messages.

Fig. 4(b) shows that between 55 and 70 messages are required on average. This is better than flooding for networks of diameter more than 4 hops; the sparse network topologies generated for our experiments typically had diameters of between 8 and 14 hops.

The number of probes sent is comparable to SENS results and is shown on Fig 4(a). Regression analysis in Table 3 shows that additional vertices correlate with additional probes, as we would expect. The coefficient of ≈ 0.5 on the Voronoi cell size variable means that we expect to use one additional probe for two marginal vertices. This is due to the cooperative nature of the Algorithm 5 where only one node attempts to verify a shared vertex.

Table 1: Average number of messages sent by TinyOS Voronoi module

Bit error rate	Mean messages sent	95% conf. range
0.00000	62.031	10.11
0.00037	62.838	10.47
0.00077	63.245	8.13

Table 2: Average number of probes sent by TinyOS Voronoi module

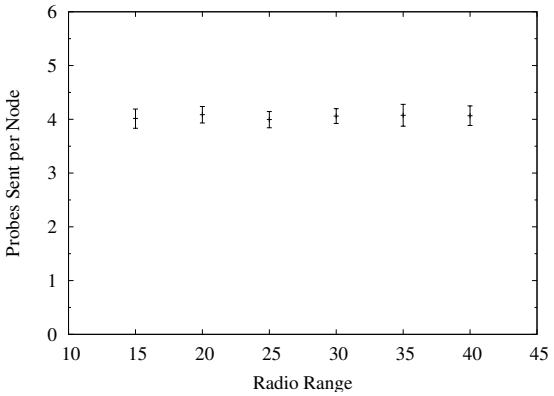
Bit error rate	Mean probes sent	95% conf. range
0.00000	4.0515	0.2844
0.00037	4.0621	0.2866
0.00077	4.0373	0.2837

Notably, neither the number of messages sent nor the number of probes is significantly affected by the bit error, as the Tables 1 and 2 show; this is an artifact of the link-layer re-transmission scheme employed by USC GPSR implementation. Due to this effect, further investigation is needed to experimentally quantify the cost of recovering from message loss at the Voronoi cell computation level.

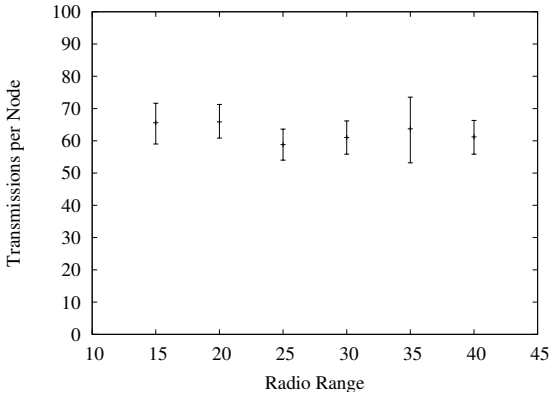
We have included a facility to track the amount of time it takes to construct a Voronoi cell. As expected, it is proportional to the final number of vertices in it and the number of probes sent, as shown by regression analysis on Table 4. There seems to be no significant correlation between number of neighbors to which node is connected and time to compute the cell; nor does the link bit error rate make any meaningful difference. The majority of nodes in our experiment took less than a minute to compute their Voronoi cells, but some outliers required several hours. We suspect that this is due to the use of very conservative inter-probe timings in our algorithm, but are still investigating this effect.

5.4.3 Limitations

The Voronoi cell computation module was developed in order to be deployed on wireless sensor nodes such as the Mica2 [15] and Telos B [19] platforms. Practical deployment on these systems was complicated by several factors. Resource requirements were problematic - although the memory usage of the Voronoi module itself is modest (2 kB RAM and 10 kB code space), it depends on the USC CLDP/GPSR module for geographic routing primitives. Since CLDP/GPSR requires an extra 2.5 kB RAM and 45 kB code space, additional optimization is required in order to deploy this system on either the Mica2, with 4 kB RAM and 128 kB code space, or the TelosB, with 10 kB RAM and 48 kB code space. The numeric precision required in order to perform repeated calculations of line intersections without accumulated round-off error posed an additional issue. The highest precision data type available on either the Mica or Telos platforms is a 32-bit float, and operations on these are very slow (about 20k ops/sec on the Mica). The USC GPSR implementation itself uses 16-bit integers for x and y coordinates, resulting in further restrictions, as the al-



(a) Average and the 95%-confidence intervals of the number of probes sent by each node



(b) Average and the 95%-confidence intervals of the number of messages sent by each node

Figure 4: Performance of TinyOS Voronoi module across topologies of varying connectivity. Five topologies for each radio range were used by both algorithms, and TOSSIM was run 20 times at 0, 0.037% and 0.077% bit error rates on each topology. The results we computed by summing number of messages sent by each node in each run and dividing by the number of nodes.

Table 3: Dependent variable: Probes Sent

Variable	Coeff.	Std. Err	t-stat	p-val
const	1.3812	0.0761	18.1597	0.0000
Vertices/cell	0.5070	0.0142	35.8812	0.0000

Mean of dependent variable	4.05028
S.D. of dependent variable	1.45827
Adjusted $\bar{R}^2$	0.152326

Table 4: Dependent variable: Time to compute Voronoi diagram (in ms)

Variable	Coeff.	Std. Err	t-stat	p-val
const	-2.8807e-06	264404.	-10.8951	0.0000
Vertices/cell	2.3214e+06	52164.0	44.5012	0.0000
Probes sent	-1.4687e-06	40173.4	-36.5580	0.0000

Mean of dependent variable	3.3920e+06
S.D. of dependent variable	5.2766e+06
Adjusted $\bar{R}^2$	0.2521
$F(2, 7157)$	1207.26

gorithm will fail if distinct vertices are mapped to the same grid coordinates. In order to avoid this, we were restricted to random topologies containing no more than about 20 nodes.

6. CONCLUSION

We have presented a fully distributed algorithm for computation of Voronoi cells in a sensor network setting. To our knowledge, it is the first method of its kind to provide construction of provably exact cells without requiring each node to contact all other nodes in the network. The results from the experiments show that this exactness guarantee has its costs, both in messages sent and time. However, for tasks requiring precise information about Voronoi cell boundaries, the algorithm proposed is a wholly feasible method, and more efficient than previously known one. To address some of the resource constraints encountered, we envision its deployment on more powerful base station nodes within hierarchical networks [10]. If lightweight geographic routing becomes available, our TinyOS module may also be deployed in smaller motes. While the memory footprint and precision of computation required may have been problematic on small systems like Mica and Telos, larger motes like Stargate [7] have ample resources to run our algorithm. We anticipate running this algorithm on a testbed of such systems in the future.

In the near future we plan on investigating other performance trade-offs by tuning implementation parameters such as timeouts and number of probes sent. Additional near-term modifications include changes to the USC GPSR implementation in order to handle higher-precision floating point coordinates, thereby addressing the problem of limited coordinate precision.

In future research, we are considering other vertex verification methods. For example, the nodes on the edge of the graph send a disproportionate number of messages while probing vertices on the boundaries of the network field; perhaps this can be mitigated by gathering additional state about locations of these edge nodes. Another area of planned exploration is the application of geographic routing to other distributed computational geometry problems, such as finding the convex hull.

Acknowledgment

The authors acknowledge the help of John Byers and Jef Considine in early version of this manuscript. The authors would also like to thank Deepak Ganesan and Gaurav Mathur for immense help in understanding TinyOS, Young-Jin Kim and Ramesh Govindan for answering many questions about the CLDP/GPSR module, David Jensen for invaluable advice throughout the project, as well as four anonymous referees for helpful comments. Boulat Bash also appreciates the support of Jim Kurose and Don Towsley in this work.

7. REFERENCES

[1] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11b mesh network. In *Proc. of SIGCOMM*, Aug. 2004.

[2] B. A. Bash, J. W. Byers, and J. Considine. Uniform random sampling in sensor networks. In *Proc. of DMSN*, Aug. 2004.

[3] J. Byers, J. Considine, and M. Mitzenmacher. Geometric generalizations of the power of two choices. In *Proc. of the 16th ACM Symp. on Parallel Algorithms and Architectures*, June 2004.

[4] B. Carbunar, A. Grama, and J. Vitek. Distributed and dynamic voronoi overlays for coverage detection and distributed hash tables in ad-hoc networks. In *Proc. of IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, July 2004.

[5] W.-P. Chen, J. C. Hou, and L. Sha. Dynamic clustering for acoustic target tracking in wireless sensor networks. *IEEE Trans. on Mobile Computing*, 3(3):258–271, 2004.

[6] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol (OLSR), October 2003. IETF RFC 3626.

[7] Crossbow Corporation. Stargate Gateway (SPB400). Available at: `http://www.xbow.com/Support/Support_pdf_files/Stargate_Manual.pdf`, 2004.

[8] M. de Berg, O. Schwarzkopf, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2nd edition, 2000.

[9] M. Demirbas and H. Ferhatosmanoglu. Peer-to-peer spatial queries in sensor networks. In *Proc. of 3rd IEEE International Conference on Peer-to-Peer Computing*, Sept. 2003.

[10] P. Desnoyers, D. Ganesan, and P. Shenoy. TSAR: A Two Tier Storage Architecture Using Interval Skip Graphs. In *Third ACM Conference on Embedded Networked Sensor Systems (SenSys).*, November 2005.

[11] D. Ganesan, S. Ratnasamy, H. Wang, and D. Estrin. Coping with irregular spatio-temporal sampling in sensor networks. In *Proc. of HotNets-II*, November 2003.

[12] R. H. Guting. An introduction to spatial database systems. *VLDB Journal*, 3(4):357–399, 1994.

[13] C.-C. Han, S. Ganeriwal, A. Boulis, and M. Srivastava. Going beyond nodal aggregates: Spatial average of a physical process in sensor networks. Poster in *ACM SenSys*, Nov. 2003.

[14] B. Harrington and Y. Huang. In-network surface simplification for sensor fields. In *GIS '05: Proceedings of the 13th annual ACM international workshop on Geographic information systems*, pages 41–50, New York, NY, USA, 2005. ACM Press.

[15] J. Hill and D. Culler. Mica: A Wireless Platform for Deeply Embedded Networks. *IEEE Micro*, 22(6):12–24, Nov/Dec 2002.

[16] B. Karp and H. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *ACM MobiCom*, Aug. 2000.

[17] Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker. Geographic routing made practical. In *USENIX NSDI*, May 2005.

[18] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: Accurate and scalable simulation of entire TinyOS applications. In *ACM SenSys*, 2004.

[19] Moteiv Corporation. *TMote Sky Datasheet*, 2005.

[20] J. A. Peterson. SENS: A simple, extensible sensor network simulator for exploring approximate aggregation techniques. Master's thesis, Boston University, 2005.

[21] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 32–43, New York, NY, USA, 2000. ACM Press.

[22] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica. Geographic routing without location information. In *ACM MobiCom*, Sept. 2003.

[23] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu. Data-centric storage in sensornets with GHT, a geographic hash table. *Mobile Networks and Applications*, 8(4):427–442, 2003.

[24] P. Samar, M. R. Pearlman, and Z. J. Haas. *The handbook of ad hoc wireless networks*, chapter Hybrid routing: the pursuit of an adaptable and scalable routing framework for ad hoc networks, pages 245–262. CRC Press, Inc., Boca Raton, FL, USA, 2003.

[25] M. Sharifzadeh and C. Shahabi. Supporting spatial aggregation in sensor network databases. In *Proc. of 12th International Symposium of ACM GIS*, Nov. 2004.

[26] I. Stanoi, M. Riedwald, D. Agrawal, and A. E. Abbadi. Discovery of influence sets in frequently updated databases. In *Proc. of 27th VLDB Conference*, 2001.

[27] USC Embedded Networks Laboratory. CLDP+GFR ver-0.2 for the motes. Available at: `http://enl.usc.edu/software.html`.

[28] Z. Zhou, S. Das, and H. Gupta. Variable radii connected sensor cover in sensor networks. In *Proc. of IEEE SECON*, 2004.