Load balanced and Efficient Hierarchical Data-Centric Storage in Sensor Networks

Yao Zhao, Yan Chen
Northwestern University, Evanston IL, USA
{yzhao,ychen}@cs.northwestern.edu

Sylvia Ratnasamy
Intel Research, Berkeley CA, USA
sylvia@intel-research.net

***Abstract*—Several new sensor network applications build on scalable, energy-aware data-centric storage. Data-centric storage is typically achieved by hashing a high-level data name to a well-known routable node address. Despite significant work on any-to-any routing for sensor networks, most schemes do not provide a node address space that is amenable to the needs of data-centric storage. Our work focuses on the problem of designing a routing primitive suitable for data-centric storage that also meets typical sensor network goals of scalability, energy-efficiency and load-balance. In this paper, we present a new Hierarchical Voronoi Graph based Routing algorithm (HVGR) that simultaneously achieves good scalability, efficiency in routing, and load balance in both routing and data storage. The region oriented routing scheme avoids overloading cluster headers by "short-cutting" routes before they actually hit cluster headers. The storage load balancing algorithm achieves uniform distribution of storage load.**

I. Introduction

Recent engineering advances in the design of small energy-efficient hardware and embedded systems have enabled large-scale sensor networks. Early sensor network data access systems were built upon the idea of data-centric routing [1, 2], while recently data-centric storage [3] was introduced and widely used for in-network storage and processing. In data-centric storage (DCS) [3], a node derives a high-level summary event from the raw data it collects; it then stores information summarizing the event at some other node that is selected based on the name of the event. This allows information about the event to be found in a scalable manner – a query specifies the name of the event and is then forwarded to the same associated node that stores the event. Further, data-centric storage has been extended to support a variety of sophisticated query primitives such as multidimensional range queries [4] and approximate queries [5].

Generally, the routing primitive of data-centric storage requires two functionalities: 1) a hash function (or other computation methods) to map a name to a meaningful label of a node in the network; 2) and the ability to route to the node that owns a mapped label. For example, in GHT [6], the label is a geographic location and the node closest to the location stores the event summary to the label. As argued in [7], data-centric storage does not have to rely on any-to-any routing. In fact, pathDCS [7] is not an any-to-any routing, because in pathDCS not all the nodes are routable. Meanwhile, many any-to-any routing algorithms are not suitable for data-centric storage. For example, it is unclear how to map a name to a high dimensional virtual coordinate or a global ID of a wireless node in many virtual coordinate based routing algorithms such as [8]–[10].

Since DCS was introduced, geographic routing is used as a routing primitive for many data-centric storage applications [4]–[6]. The dependency on GPS or other location devices is an extra limitation, and it may not be applicable in some sensor network. Therefore, it is desirable to remove the requirement of location devices. Meanwhile, data-centric storage solutions in sensor networks must be self-organizing, scalable (low per-node state), efficient (low routing stretch and route construction overhead) and load-balanced (for robustness and to avoid energy depletion). A desirable routing primitive for data-centric storage should meet these challenges.

In this paper we present a routing primitive for data-centric storage called hierarchical Voronoi graph routing (HVGR) that simultaneously achieves good scalability, efficiency and load balance. Hierarchies are desirable for the scalability they offer, and hierarchical addressing is nature for the mapping of events in data-centric storage [4, 5]. However, applying hierarchical routing to large, self-organizing and resource-limited sensor networks is non-trivial for three reasons:

- **Efficiency problem**: Hierarchical routing may elongate the routing path. The challenge is designing the hierarchy and the routing without global information that can conduct close to shortest path routing on it.
- **Load balancing in routing**: A hierarchical routing for sensor networks should avoid overloading the "leaders" in the hierarchy. For example, the hierarchy routing [11] from ad-hoc networks does not fit our requirements as it overloads the cluster headers, quickly depleting their energy.
- **Load balancing in storage**: Storage load balancing is very important in DCS because of the primary power constraint and limited storage space of sensor nodes. Balanced hierarchical addressing decomposition is a key issue to ensure load balanced storage in data-centric storage.

HVGR uses concepts from Voronoi graphs to construct and maintain a virtual hierarchy that spans all sensor nodes and does so in a manner that is self-organizing and does not require the use of GPS or other geo-location devices. The region oriented routing is specially designed for the hierarchy to avoid overloading landmarks by "short-cutting" routes before traffics actually hit landmark nodes. Meanwhile, the region oriented routing achieves efficient routing paths closing to shortest paths in practice. A simple and practical hashing scheme is designed to achieve balanced storage on every node in the network potentially, which circumvents the hard problem of uniform hierarchical decomposition. In summary, HVGR represents a practical protocol with the following features:

- **Good Scalability.** HVGR is a scalable virtual coordinate based routing scheme which does not require any location

information. Unlike other virtual coordinate approaches [8, 9, 12, 13], route dead ends do not exist in a stable HVGR system. The initialization overhead is $O(N \log N)$ and every node maintain $O(\log N)$ routing states, where N is the number of nodes in the sensor network.

- **Good Efficiency.** Routing paths in HVGR are close to optimal, *i.e.*, the routing stretch (length of the routing path divided by the length of the shortest path between two nodes) is close to one in average on all paths and locally short paths. Furthermore, we borrow the routing idea from the virtual coordinate routing [8, 14] to further reduce the routing stretch.
- **Good Load Balancing for Routing.** Although HVGR uses a hierarchical virtual coordinate, it avoids the classic problems of hierarchical approaches such as bottlenecks at root, overloading the backbone structure or gateway nodes. HVGR achieves this through its use of region oriented routing which does not introduce any backbone routes or hotspots.
- **Good Load Balancing for Data Storage.** HVGR achieves balanced storage load, which means no particular node will be overloaded or run out of storage space quickly. In addition, when a node plans to sleep proactively, it need only transfer the stored data to its neighboring nodes instead of some distant virtual neighbors (as in VRR [15]).

We simulate HVGR extensively and compare it to VRR [15], a state-of-the-art routing protocol for sensor networks. Our evaluation shows that HVGR successfully achieves the above design goals. The remainder of this paper is organized as follows: we survey related work in Section II, present the details of our HVGR algorithm in Section III, evaluate HVGR in Section IV and conclude in Section V.

II. Related Work

A vast number of sensor networks comprise static sensor nodes. Our work focuses on these scenarios of $large$, $static$ sensor networks with varied node $density$. While some deployment scenarios do involve mobility in which sensors are carried by human or animals, we do not consider these in this paper.

a) Data-centric storage based on geographic routing: Geographic routing is a class of scalable any-to-any routing in wireless networks [16], and geographic hash table (GHT) [6] first proposed data-centric storage by extending the idea of geographic routing. GHT is simple and light-weight but inherits the shortcomings of geographic routing in that it requires GPS (or other position system) and suffers from routing dead ends in sparse networks. Moreover, for data-centric storage, GHT requires a priori knowledge of the network's boundary and it is not clear how to uniformly distribute storage for irregular boundaries. Following research such as DIM [4] and DIFS [5] further improves the functionality of the data-centric storage for complex query primitives. Also KDDCS [17] improves the storage load balancing on DIM [4] so that the storage of events is distributed reasonably uniformly among the sensors.

b) Routing for data-centric storage without out GPS: PathDCS [7] is designed for data-centric storage. It is simple and practical. But in pathDCS, not all the nodes in the network can be storage servers, and hence load balance of the storage is not good. Each routing path must go through at least one landmark, hence landmarks are prone to be overloaded. Therefore we do not specifically compare our approach with pathDCS in evaluation. VRR [15] is another candidate for data-centric storage because of the DHT abstraction it provides. VRR however requires $O(\sqrt{N})$ routing state at every node, which is not scalable for large sensor networks. Since VRR's virtual ring topology is independent to the underlying topology, VRR works quite well in mobile scenario, while the path stretch is relatively large. GEM [18] scheme is another scalable routing solution for data-centric storage. GEM builds a virtual polar coordinate system and an event is hashed to a distance and angle in the polar coordinate space. However, GEM is fragile under dynamics – when nodes/links fail, a potentially large number of nodes in the system must recompute routing coordinates.

c) Any-to-any routing algorithms that are not suitable for DCS: Recently there are a flourish of landmark or virtual coordinate based any-to-any routing in sensor networks. Rao et al. propose a scheme for geographic routing without location information (NoGeo) [19]. They present a virtual coordinate construction algorithm, which achieves performance comparable to that of real geometric coordinates in dense networks. However, NoGeo requires $O(N)$ per-node states and causes $O(N\sqrt{N})$ communication overhead during initialization. NoGeo stimulates several other solutions that construct virtual coordinates [8, 9, 12]–[14] for efficient any-to-any routing. S4 [20] is a novel landmark based routing which guarantees routing success, and bounds the path stretch to be 3 in the worst case. However, in these virtual coordinate or landmark based routing, an additional scalable location service is required. Moreover, it is unclear how to hash an event to meaningful virtual coordinates because large portions of the high dimensional virtual coordinate space are likely to be unoccupied. It is thus difficult (or at best costly) to find the node closest to a coordinate derived by hashing an event name. In summary, these any-to-any routing approaches cannot be applied to data-centric storage in practice until they themselves can provide a name service.

d) Related hierarchical routing: The idea of hierarchical routing is also widely used in the point-to-point ad hoc routing [11, 21]–[23]. CGSR [21], LANMAR [22] and ZRP [23] simply use two level of hierarchy to improve the scalability and HSR [11] will potentially overload cluster headers by building paths over cluster headers. Hence these existing hierarchical routing can not be simply borrowed for data-centric storage. Actually, one of the papers that influence this work is that of Tsuchiya [24], which introduced a classic hierarchical routing in wired networks.

III. Hierarchical Voronoi Graph Routing

A Voronoi graph is a well-known decomposition of spaces or planes [25]. Given a set of landmarks, a Voronoi graph divides the plane according to the nearest-neighbor rule: each point is associated with the region of the landmark closest to it. Our hierarchical virtual coordinate construction is analogous to the decomposition of a Voronoi graph, which inspires the name, Hierarchical Voronoi Graph Routing (HVGR). In what follows, we first present the basic HVGR routing algorithm followed by the practical issues facing this basic scheme and our solutions to these challenges.

A. Basic Routing Algorithm

We first introduce our hierarchical coordinates and then describe our region oriented routing that forwards packets to specified hierarchical coordinates. Finally, we show how name-based routing for data-centric storage is achieved in HVGR.

1) Features of hierarchical coordinates: The basic routing algorithm relies on the hierarchical coordinates associated with landmarks. To begin, there are some number of first level (or highest level) landmarks which divide the network into different first level subregions. Fore example, in Figure 1, L_1 is a first level landmark and the square [1] marked *1^{st} level* is L_1's first level region. Each node in the network knows all the first level landmarks and its next hops to these first level landmarks. Then, within each first level region there are some second level landmarks (*e.g.* $L_{1,1}$ in L_1's first level region), that divide the first level region into multiple second level subregions. Again, each node in a particular first level subregion knows the path to all second level landmarks that lie within its first level subregion. This hierarchical division continues until the nth subregion is sufficiently small and, finally, every node knows all other nodes within its lowest level subregion. A node's hierarchical coordinate is a sequence of landmark node identifiers, ordered from the highest to lowest level. For example, in Figure 1, node D has the hierarchical coordinate $(L_1, L_{1,1}, L_{1,1,1})$.

2) Region oriented routing to a hierarchical coordinate: Figure 1 shows an example of the basic routing algorithm used to route packets from the source (S) to a destination (D) with known hierarchical coordinate. The first level landmark of D is L_1, which is easy to determine given D's hierarchical coordinate. Our first step sends the packet towards L_1, to which the routing path is known by all nodes in the network. The packet is forwarded hop by hop towards landmark L_1 until it enters the first level region (say, region R_1) of L_1. Assume the packet enters region R_1 of L_1 through node A. Node A knows all the second level landmarks in region R_1 including $L_{1,1}$. Hence A will similarly forward the packet toward $L_{1,1}$ (the second level landmark in D's coordinate vector). The packet will change direction when it reaches node B, which is the entry to the second level region within which D lies. Finally the packet reaches the lowest level subregion whose entry node knows the direct path to D (because we require complete routability in the lowest level regions).

In many classical hierarchical routing algorithms, landmark nodes (or cluster heads) are prone to be overloaded. This is avoided in HVGR because, although the region oriented routing uses landmarks to guide routing, packets are redirected when they enter the regions of the landmarks. Also, there is no special "gateway" node in our hierarchical system as packet may enter a region through any border node of the region. It is worth mentioning that the routing path between two close nodes in different regions may not be optimal, as the initial direction of the region oriented routing may be away from the direct path in certain cases. However the detour is locally bounded and, as we show (Section IV-B), HVGR has small path stretch for paths between close nodes.

[1]The shape of a region actually can be irregular as we split the regions using Voronoi graph described in Section III-C.

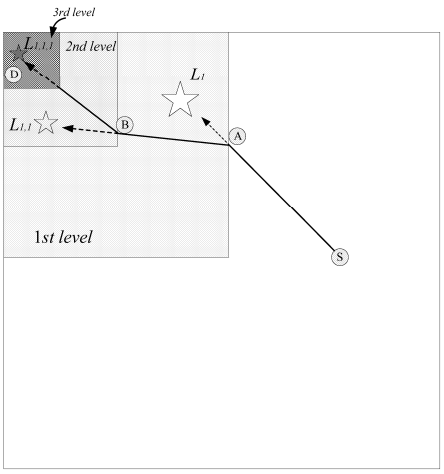

Fig. 1. Region oriented routing.

3) Basic name based routing for data-centric storage: In data-centric storage, events are generated and stored at nodes corresponding to event names. Assume now we have a new event with event ID E, and there are m_i i^{th} level landmarks in an $(i-1)^{th}$ level subregion[2]. We introduce a series of hash functions H_i $(i = 1, 2, ...)$ for each level of the hierarchy (or a single hash function with multiple seeds). First, E is hashed to an integer $l_1 \in [1, m_1]$ with hash function H_1. Therefore we determine that the event E is stored in the first level region of the l_i^{th} first level landmark, say L_{l_1}. Here we assume the order of the first level landmarks is fixed and known to every node. Then the event is forwarded towards landmark L_i, as in region based routing. Once the event enters the right first level region, the entry node uses hash function H_2 to select the next second level landmark for the event. This iterative process creates a hierarchical coordinate *while the event is being forwarded*. Finally the event enters the lowest level region wherein all nodes know each other and hence the event can be simply stored at the node with ID closest to the hash of event ID E.

Our name based routing has an important advantage for name-based storage: to withstand node failure or if a node proactively chooses to sleep, the node just needs to replicate or transfer the stored data to its neighboring nodes in the same lowest level region. This is more efficient than other DHT based storage system, *e.g.*VRR [15], which has to transfer the data to remote nodes (because VRR's virtual neighbors may not be topologically close). We leave for future work the design of sleeping strategies optimized for our HVGR system.

B. Practical Design Issues

There are several challenges to building a scalable, efficient, robust and load balanced data-centric storage system over HVGR.

The first issue is that of efficiently building a scalable hierarchical virtual coordinate system. Tsuchiya proposed hierarchical landmark routing for the Internet which yields an average routing table size of $O(\sqrt{N})$ [24]. But intuitively, we expect an average routing table size of $O(\log N)$.

The second challenge is to balance load across the system. In this paper, we consider load balancing for both routing and

[2]The 0th level subregion refers to the entire network

storage. With good load balancing in routing, no node becomes overloaded for forwarding packets. HVGR's routing achieves this by using landmarks as routing guides but not actually forwarding packets through the landmarks. In terms of storage load, the goal is to have all nodes to store similar amounts of data. We present certain improvements to our basic name-based routing in order to achieve load-balanced storage.

The third challenge is to achieve a path stretch close to 1. The region oriented routing achieves about 1.15 path stretch over shortest path. While this is a useful improvement over VRR [15], there is still room for even better efficiency.

The final challenge is to provide resilience against dynamic changes in nodes and links. For example, new nodes may join while other nodes may proactively sleep or just fail without notification. It is desirable that the system deals with such dynamics in a fast and efficient manner.

In what follows, we address each of the above challenges in turn. We introduce algorithms for hierarchical virtual coordinate construction, landmark selection, storage assignment, path stretch improvement and robustness to network dynamics.

C. Constructing the Hierarchical Architecture

To construct a hierarchical architecture, there are two general approaches: bottom-up and top-down. Some hierarchical routing protocols in ad hoc networks take the bottom-up approach, *i.e.*, constructing the lowest level first and then building upper levels iteratively. For example, Hierarchical State Routing (HSR) [11] is a multilevel clustering-based routing protocol. This bottom-up approach typically generates a backbone path between landmarks or gateway nodes between regions. In this paper, we introduce a top-down approach to support the name based routing introduced in Section III-A.

We will present our landmark selection algorithm in Section III-D. For now, we assume a landmark selection algorithm that selects at most m_i $(i = 1, 2, \ldots)$ landmarks in an $(i-1)^{th}$ level region. Using the example in Figure 2, we now present the details of our hierarchy construction.

1) First, assume m_1 first level landmarks are selected. Each landmark node floods a LANDMARK packet to the entire network and thus every node learns its distance to all first-level landmarks (See Figure 2 (a)).
2) Each node chooses the closest landmark as its representative (ties are broken in a consistent manner, *e.g.*, based on the ID of the landmark nodes). The entire network is thus divided into m_1 localized subregions. An important feature of this division is that each region itself is a connected sub-network. This is because if a node belongs to the region of landmark L, its parent node in L's shortest path tree must also be in the region of L (See Figure 2 (b)).
3) Next, consider a first level region R of a first level landmark. The m_2 second level landmarks in region R do a *scoped flood* to only the nodes in the first level region R. The scoped flood is controlled as follows: a LANDMARK packet generated by the second level landmark has a field of region, which can be the ID of the corresponding first level landmark. A node will rebroadcast the received LANDMARK packet if and only if the node is in the region R. There are $m_1 \times m_2$ second level landmarks in the entire network. However the overhead of scoped flooding is only $m_2 N$ in total because of the scoped flooding. Therefore, the total overhead due to LANDMARK packets is equivalent to m_2 network-wide floods (see Figure 2 (c)).
4) The second level landmarks divide the first level regions into some second level subregions (*e.g.* shown in Figure 2 (d)). By doing this iteratively, nodes' hierarchical virtual coordinates are created. The termination condition we use is that the subregion only contains the owner landmark and the landmark's one-hop neighbors. Therefore, it is easy for a node in the lowest region to find routes to any other node in the region. Note that it is also possible to allow larger lowest level subregions and use any all-pairs routing algorithm (*e.g.* link state or distance vector based routing) within the region.

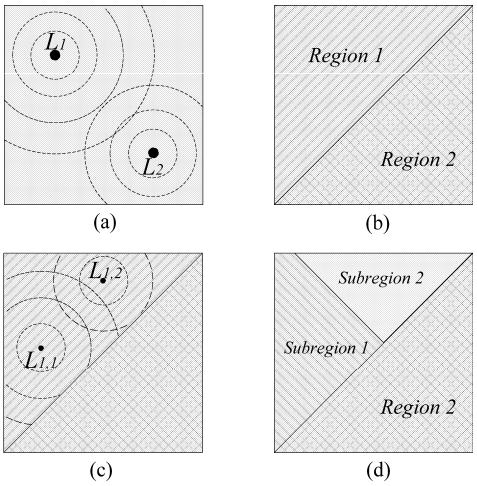

Fig. 2. Hierarchical virtual coordinate.

Intuitively, after $O(\log N)$-level divisions, we obtain the lowest level regions. While our simulation results support this intuition for the sensor network topologies we consider, a rigorous proof of this is non-trivial and we leave for future work. The flooding overhead of LANDMARK packets of each level is $O(m_i N)$ and hence the total flooding overhead of all the levels is $O(mN \log N)$, where $m = \max_i m_i$.

D. Landmark Selection

Landmark selection is a critical problem for all landmark (or beacon) based routing algorithms [8, 12, 13, 20]. However, there are few studies exploring the landmark selection problem. One possible reason is that complex landmark selection algorithms (*e.g.* using global information or multi-round election) are hard to support in sensor network scenarios. Also, the exact landmark selection criteria favoring the routing is not clear in most routing algorithms and random landmark selection seems to do an enough good job.

Intuitively, we want the landmarks in a region to be dispersed so that the subregions divided by them are even. Randomly selecting landmarks is a simple but non-optimal approach to uniformly spreading out landmarks. An optimized random landmark selection algorithm is introduced in [14] to reduce the chance of picking nearby landmarks. The basic idea in [14] is to first build a shortest path tree rooted at a coordinator node by flooding. Then starting at the leaves nodes, each node collects

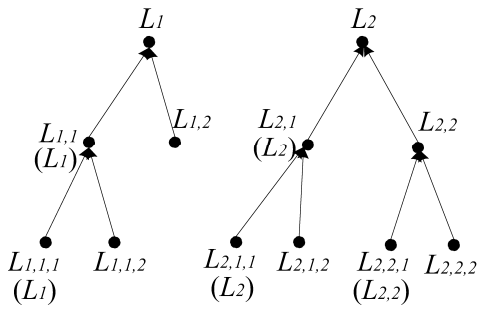

Fig. 3. Landmark hierarchy.

the landmark candidates from its child nodes in the shortest path tree, and randomly picks a certain number candidates to report to its parent node. Assuming N_R nodes in a region, then the overhead of landmark selection in this region is $O(2N_R)$. A useful by-product is that the coordinator learns the number of nodes in this region (*i.e.* N_R); a feature we leverage in Section III-E.

We now introduce how landmarks are selected at each level. First, assume there is a node C which starts the landmark selection process [3]. Node C automatically becomes a first level landmark as well as the *master* landmark among all first level landmarks. Using the optimized random landmark selection algorithm [14], node C selects the other $m_1 - 1$ landmarks and divides the network into m_i first level regions. Each of these m_1 landmarks automatically becomes the master landmark of its first level region. Then each master of its region repeats the process of random landmark selection. Finally, a master landmark stops selecting slave landmarks when it finds that all the nodes in its region are its neighbors. Note that in an $(i-1)$th level region, maybe less than m_i landmarks are finally selected because the region is already very small. Figure 3 shows an example where landmarks create a hierarchical structure. For example, using our landmarks selection algorithm, landmarks L_1, $L_{1,1}$ and $L_{1,1,1}$ are the same node.

E. Storage Load Balancing

Balancing load across nodes in a sensor network is critical as sensors are energy and storage constrained and energy-depleted nodes can result in network partitions. The storage load balancing goal here is thus to uniformly distribute events to nodes.

If we use the basic name based routing introduced in Section III-A, the storage load balancing property depends on whether the landmarks divide the networks evenly or not. For example, assume the m_1 first level landmarks are $L_1, \cdots, L_{m_1}$. An event E has a probability $1/m_1$ of being assigned to the first level region of landmark $L_i (i = 1, \ldots, m_1)$. If L_1 has a larger region with more nodes than L_2, then in average a node in L_1's region will have less stored events than those in L_2's region. Unfortunately, it is hard to divide a region evenly with a simple random landmark selection algorithm. Landmarks may have to negotiate and adjust their divisions, and hence make a complex and costly process. We solve this challenge through a slight modification to the basic name based routing.

[3]Node C do not have to be any special node in some special location (*e.g.* at the center). We choose a random node C in our simulations and the results do not vary much with different C nodes.

As mentioned in the landmark selection algorithm (Section III-D), each landmark knows the number of nodes in its region. We thus assign storage tasks to regions in proportion to the sizes of the regions. For example, for the m_1 first level landmarks, let $N_i (i = 1, \ldots, m_1)$ be the number of nodes in landmark L_i's first level region. So $N = \sum_{i=1}^{m_1} N_i$. H_1 is a hash function which hashes an event ID to a real value in $[0, 1)$. Then the event will be stored in a node in L_k's first level region if and only if:

$$\frac{1}{N}\sum_{i=1}^{k-1} N_i \leq H_1(E) < \frac{1}{N}\sum_{i=1}^{k} N_i \quad (1)$$

For example, assume we have three first level landmarks, and $N_1 = N_2 = 100, N_3 = 200$. If $H_1(E) < 0.25$, the event is assigned to the region of landmark L_1; if $H_1(E) \geq 0.5$, the event will be assigned to the region of L_3; otherwise the event will be stored to the L_2's region. Since the hash function H_1 is uniform, each region is expected to store a number of events proportional to the number of nodes in that region. When an event is assigned to a first level region, it is iteratively hashed to a second level region and so on. In the lowest level region, each node can be viewed as a subregion and be assigned the hash range evenly by the landmarks.

F. Path Stretch Reduction

Our region oriented routing uses only a single landmark to guide the forwarding at any point in time. However, one might leverage the information provided by the other landmarks. For example, our region oriented routing could be combined with other single-layer landmark based routing algorithms such as HopID [14] and BVR [8].

For simplicity, consider only the first level landmarks. Each node knows their shortest path distance (or hop distance [14]) to the first level landmarks[4], and a vector of such hop distance in a certain order makes the coordinate of the node, *i.e.*, the Hop ID coordinate. Using the Hop ID as the destination coordinate, greedy algorithms can be used to route efficiently [8, 14]. Dead end problem can be completely solved because our region oriented routing can help out the dead ends. In short, if we know the hierarchical virtual coordinate and the Hop ID coordinate of the destination node, we first conduct the greedy routing based on the Hop ID coordinate. Once the packet is stuck at a dead end, the region oriented routing can be used to lead the packet out of the dead end. The greedy Hop ID routing resumes when the region oriented routing brings the packet to a node "closer" to the destination than the dead end. We call the combined routing scheme *HVGR+*, to differentiate it from our original scheme HVGR. Section IV-B shows that HVGR+ improves HVGR significantly in terms of path stretch.

In data-centric storage, the generator of an event or the querier does not initially know the coordinates of the storer and hence HVGR+ can not be directly applied. However a flow usually involves back-and-forth communication between two end nodes and hence the coordinates of the source or destination can be piggy-backed and HVGR+ can thus be applied for all but the first packets in a flow.

[4]In case there are very few first level landmarks; *e.g.* two landmarks, we also use the distance to second level landmarks.

G. Handling Dynamic Changes

Robustness is desirable for data-centric storage systems. We now introduce schemes that dynamically adapt the hierarchical virtual coordinates under node and network dynamics. In a nutshell, our goal is to maintain the shortest path trees to landmarks in the corresponding regions. An individual node seeks to maintain the routes (or next hops) to $O(\log N)$ landmarks in the regions within which it lies. This involves maintaining multiple distance vector routing paths and hence techniques from existing distance vector based routing solutions can be applied here. Given the similarity and due to space limitations, we present only a high-level description of our coordinate maintenance algorithms in this paper. An additional design guideline we follow is to minimize changes in nodes' hierarchical coordinates so that the number of stored events that need to be relocated under dynamics is small.

1) Periodic HELLO *messages:* Every node periodically broadcasts HELLO messages. HELLO messages are used for neighbor maintenance and landmark-rooted shortest path tree maintenance. A HELLO message includes: 1) the node's hierarchical virtual coordinate as well as the hop distance to the corresponding landmarks, 2) the latest *sequence numbers* for the landmarks obtained by listening to neighbors' HELLO messages. Each landmark maintains a counter for its sequence number, and the sequence number is increased by one and broadcast to its neighbors every time the landmark sends out a HELLO message. Using these HELLO messages, the latest sequence number of an ith level landmark is gradually flooded to the $(i-1)$th level region containing the landmark. This looks like a slow flooding from the landmark to the right region. The sequence number helps the fast recovery of routes to landmarks when node/link failures happen (See Section III-G3). Note that in standard distance vector routing sequence numbers are not used because it is not possible to include $O(N)$ sequence number in the HELLO messages.

If a node does not hear any HELLO messages from a neighbor for a certain period, this neighbor is considered failed and removed from the node's neighbor table.

2) Node join: When a new node A joins the sensor network, it learns its neighbors' virtual coordinates from the HELLO messages of its neighbors. Node A first picks a lowest region that has the closest landmark to itself, and then informs the landmark (say landmark L) about its join. If A is one hop away from L, landmark L simply broadcast a message to tell all the other nodes in the lowest region (note all the nodes in a same lowest level region are within one hop from the lowest level landmark); if A is two hops away from L, node A will becomes a new landmark, following our approach of the construction of hierarchical virtual coordinates in Section III-C. In both cases, the communication overhead of a new node's join is $O(1)$, because the lowest level region has $O(1)$ nodes. Later on, node A learns its parent nodes to the landmarks of different levels that are visible to it by listening to the HELLO messages.

3) Node failure or link failure: Usually, node failure and link failure are hard to differentiate. So once a neighbor is unreachable, we assume that the neighbor node fails.

a) Landmark failure: Although landmarks play an important role during initialization, they are not as critical once the coordinates are constructed. Region oriented routing can continue to work even if some landmarks fail because a routing path may not pass through the failed landmarks at all. We can thus adopt a slow but consequently light-weight approach to recover from landmark failures.

Note that in HVGR, landmarks also build a hierarchical structure (See Figure 3). With the exception of the master landmark at the first level, every landmark is a slave landmark at some level. A simple scheme is to let the master landmarks periodically check whether the slave landmarks are alive or not. However, this scheme increases the overhead on landmarks. Instead, we propose a light-weight scheme the leverages the sequence numbers in HELLO messages. Normally, in any region, the master landmark should periodically receive the new sequence numbers updated via HELLO messages from the other landmarks in the region. Once the master landmark does not hear new sequence number from a slave master for a certain time, the master landmark sends a query to the slave master to check if the slave master is still alive. If the slave master is unreachable, the master landmark will select a node in the slave master's subregion as replacement. The new landmark can immediately do a scoped flood of the region to claim its replacement or let HELLO messages gradually announce the replacement. As for the first level master landmarks, we let the other first level landmarks also monitor its availability. If the master landmark fails, another master landmark is elected among the slave landmarks using classic ring election or bully algorithms [26]. The new master landmark then selects a new landmark for replacement the former master landmark.

b) Non-landmark node failure: If a non-landmark node fails, its neighboring nodes may lose the next hop to some landmarks. This is the same route recovery problem as in distance vector based routing algorithms. To deal with the well-known "count-to-infinity" problem [27], we again leverage the sequence number carried in HELLO messages. When a node A finds its next hop node to a landmark L fails, it can simply choose another neighboring node that broadcasts newer sequence numbers for the landmark. This scheme is light-weight, as the node losing paths simply waits for neighbors' HELLO messages.

4) Storage Hash Range Adjustment: For data-centric storage, each region is responsible for a certain hash range so that the event associated with the hash value in the range is stored in this region, as shown in Section III-E. When nodes join or leave, the hash range may need to be adjusted to keep load balanced. However, a node's join or leave without any action usually does not hurt the load balancing property much. To save the control overhead, we decide to not adjust the hash range of regions when detecting the events of new or failed node. On the other hand, landmarks infrequently collect the number of nodes in its region and check if there are significant changes or not. Once the number of nodes in the region changes significantly (*e.g.*, increase or reduce more than 20%), new hash ranges are negotiated by landmarks and distributed to nodes in the corresponding region.

IV. EVALUATION

A. Evaluation Methodology

1) Simulator: Similar to [8, 9, 14], we first implemented a packet level simulator in C++ that can scale to tens of thousands of nodes. In this simulator, radios have a precise

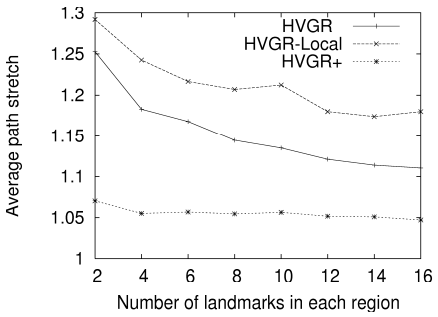

Fig. 4. Path stretch *vs* # of landmarks in each region.

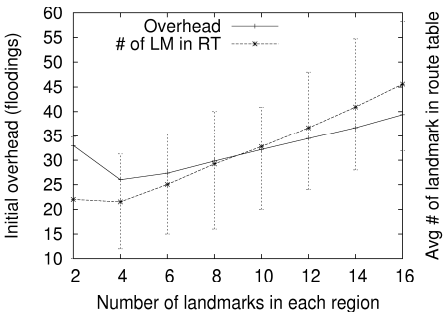

Fig. 5. Initialization overhead & route table size.

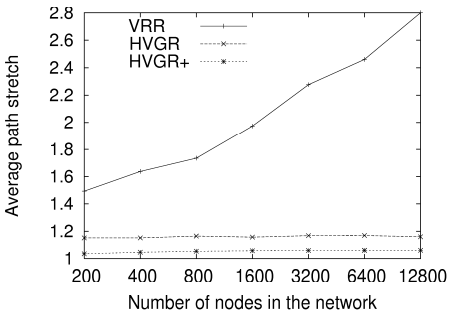

Fig. 6. Path stretch *vs* network size.

(circular) radio range 1, and nodes can send packets only to nodes within this range. There is no bandwidth limit in the simulator. We do not simulate collision in the simulator. While these assumptions are not very realistic, the simulations empirically verify the correctness and the feasibility of the protocols. It is our future work to implement HVGR in some sensor network testbeds. For comparison, we also implemented VRR [15], the latest wireless routing protocol which also works for data-centric storage.

2) Simulation Setup: Our simulation scenarios include N nodes distributed randomly in a 2-D $C \times C$ square area. The network density (denoted as λ) is defined as the average degree of the nodes. Unless otherwise indicated, we choose N as 3,200 and $\lambda = 3\pi$ in the baseline simulations. We randomly pick 200 nodes as the sources and generate 200 random events to store in the system. For each configuration, we conduct 10 random runs and report the aggregate statistics.

3) Metrics: The following metrics are considered.

- *Path stretch*: the ratio of the real routing path length to the shortest path length between the source and destination. Path stretch is a common metric used in [8, 14, 15, 18] to evaluate the routing efficiency.
- *Load balancing*: Entropy or divergence to the uniform distribution can be used to estimate whether loads are distributed uniformly or not. However, we find it is clearer and more intuitive to use the cumulative distribution function of the loads on nodes.
- *Route table size*: The route table size is proportional to the number of landmarks visible and stored in the route table.
- *Initialization overhead*: the number of packets sent to build the hierarchical virtual coordinate. For simplicity, we use the unit of *flooding*. If M packets are sent, we say the overhead is M/N floodings, as a flooding to the whole network costs N packets.
- *Maintenance overhead*: the number of packets sent when a dynamic event (*e.g.*, node join or failure) happens.

Next, we will present the simulation results.

B. Selection of Number of Landmarks in Each Level

The number of landmarks is critical to the performance of any landmark based routing algorithms [8, 9, 14, 20]. Generally speaking, the more landmarks the less path stretches and the higher routing success rate, and also the more overhead. However, in our HVGR, the number of landmarks is not so critical. If there are less landmarks in each level, then there will be more levels of division. HVGR+ relies on the first level landmarks to conduct Hop ID routing [14], and hence the number of first level landmarks affects the performance of HVGR+. For simplicity, we let m_i, $(i = 1, 2, \ldots)$ be the same and equal to m.

We vary m in the baseline simulation setups. Figure 4 shows the path stretch as a function of different m values. For HVGR, path stretch drops quickly as m increases when m is very small. When m is larger than 8, the speed of path stretch reduction is very small as m increases. On the other hand, HVGR+ does not change remarkable as m varies. The path stretch of HVGR is usually about 1.15, while that of HVGR+ is about 1.05, both close to one. Sometimes it is quite reasonable to have highly localized traffic, especially for a large-scale network. Therefore we also evaluate the path stretch for all the path of length between 2 and 5 hops in shortest path routing. We take such paths to represent local communication as the network diameter is more than 30 hops and delivery of data to one-hop neighbors is trivial. In Figure 4, HVGR-local shows that the path stretch for local communication is still very small (less than 1.3).

Figure 5 shows the relationship between m and the initialization overhead. It seems that the initialization overhead increases linearly as m increases except the special case of $m = 2$. Figure 5 also shows that the minimal, average and maximum number of landmarks in nodes' route tables increase slowly as m goes up when $m \geq 4$. When $m = 2$, the chance of unbalanced partition of Voronoi graph by the randomly chosen landmarks is much larger, which causes the special case.

From Figures 4 and 5, we can tell that the number of landmarks in each region is not critical to the performance of HVGR and HVGR+. In the following simulations, we choose m as 6, considering the tradeoff between path stretch and initialization overhead.

C. Scalable to Large-Size Sensor Networks

To study the scalability, we vary the number of nodes from 200 to 12,800 in the sensor networks, while keeping the same density (*i.e.*,$\lambda = 3\pi$) by increasing the area.

Figure 6 shows that the path stretches of both HVGR and HVGR+ do not increase remarkably as the network becomes significantly larger. Note that x axis is in logarithmic scale. When the sensor network has even 12,800 nodes, the path stretch of HVGR is still less than 1.18 and that of HVGR+ is about 1.06. The reason of stable path stretch of HVGR lie in the nature of hierarchical structure. Recalling the region oriented routing, the path is longer than shortest path as the packet is forward towards the closest landmark to the destination. The deviation angle of the direction may determine the path stretch,

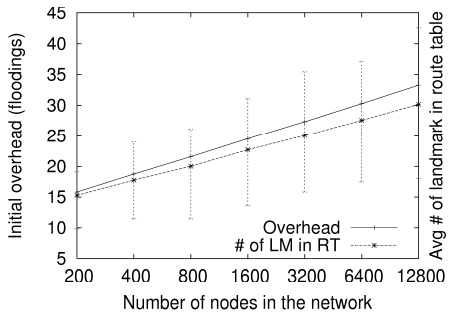

Fig. 7. Initialization overhead and route table size *vs* network size.

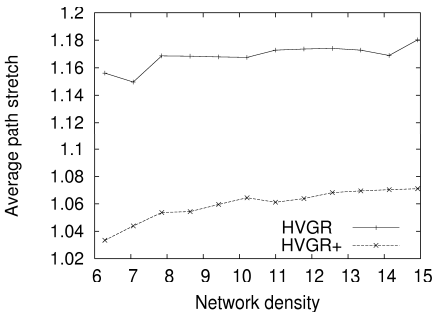

Fig. 8. Path stretch *vs* network density.

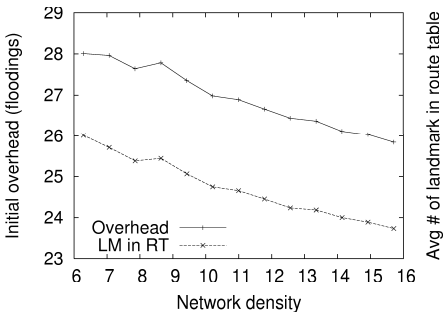

Fig. 9. Initialization overhead and route table size *vs* network density.

but the angle seems to be independent to the network size. Clearly VRR has much larger path stretch, especially in large networks. The path stretch of VRR is above two when the network has more than 2,000 nodes.

Figure 7 shows that the average number of landmarks in nodes' route table goes up linearly as the number of nodes increases exponentially, which confirms our conjecture mentioned in Section III-C. Consequently, the route table size of each node and the initialization overhead of HVGR is linear to $\log N$, shown in Figure 7. Even when the network has as many as 12,800 nodes, the route table contains about 31 landmarks in average. Meanwhile, the maximum route table size of all the nodes is normally less than twice of the average table size (See the errorbars in Figure 7). This result clearly shows that our HVGR is scalable to extremely large sensor networks. On the other hand, VRR requires $O(\sqrt{N})$ memory for the route table, which is less scalable.

D. Efficient with Networks of Various Density

Usually (virtual) coordinate based routing protocols [8, 14, 16] perform worse in sparse networks than they do in dense networks because there are more dead ends in sparse networks. Our HVGR is pure topology based and has no dead end problem, hence network density will not affect the performance of HVGR much. To evaluate this, we simulate with networks of 3,200 nodes with various network density. The density varies from 2π to 5π, which covers most practical network densities. Sparser networks with $\lambda < 2\pi$ usually have low connectivity and are partitioned into fragments [28].

Figure 8 shows that HVGR and HVGR+ have a little bit larger path stretches in dense network than those in sparse networks. This seems to be counter-intuitive, as we do not expect HVGR and HVGR+ perform better in sparse networks. The reason may be that the path stretch is a relative factor. Obviously the denser the network, the smaller the average length of routing paths. Therefore, the path stretch of dense networks is larger than that of sparse networks given the same extra hops over the shortest path. Figure 9 shows the initialization overhead drops as networks become denser and nodes are confined in smaller areas. This is reasonable as the average number of levels of hierarchy decreases when the area of sensor networks shrinks.

E. Balanced Load for Both Routing and Data Storage

Now we study the load balancing feature of our routing algorithms. First, we check the routing load balancing for HVGR, HVGR+ and the ideal shortest path routing. We count the number of forwarded packets of each node during the simulation of the routing algorithms and then normalize them so that on average a node forwards only one packet. We do the normalization because different routing protocols have different routing paths length even when the networks and source-destination pairs are the same. Figure 10 shows the cumulative distribution function of the forwarding load of the nodes under different routing protocols. Basically, we can see that HVGR, HVGR+ and the idea shortest path routing have similar routing load balancing property. This tells that *HVGR and HVGR+ do not overload landmark nodes*, as we expected.

Meanwhile, we notice that the forwarding loads are quite unbalanced for all the three algorithms. This is because the nodes in the center of the area usually have large forwarding loads, while the nodes in the edge of the network may seldom forward packets for other nodes. Generally speaking, minimizing the path length renders the overload of the nodes in the center. There is a tradeoff between the path stretch and the routing load balancing. We will study this interesting tradeoff in the future.

Second, we consider the storage load balancing. We generate 12,800 events and let the data-centric storage systems store them. In an ideal hash based storage system, an event ID is uniformly hashed to a node, and hence the number of events stored on a node has a binomial distribution. Figure 11 shows the CDF of the numbers of events stored in the 3,200 nodes when we simulated an ideal hash based storage system, HVGR and VRR. HVGR has the perfect storage load balancing, as the two lines of the ideal system and HVGR are overlapped. More than 90% nodes have the load within the range from 10 to 20 events, while the average is 16. On the other hand, VRR has worse load balancing property. This is because the node IDs of VRR do not uniformly split the ID space, although the IDs are hashed randomly. If three node IDs are by chance very close, then the node with the middle ID shall get few stored events.

F. Small Overhead for Dynamic Scenarios

In this section, we evaluate the maintenance overhead of our HVGR system. As we mentioned in Section III-G1, HELLO messages are broadcasted periodically by every node, hence we do not count HELLO messages here. Meanwhile, we do not re-assign the hash range of each region unless the system is very unbalanced (See Section III-G4). Therefore, we focus on the overhead caused by nodes' join and failure to the hierarchical virtual coordinate here. In a 3,200-node network, we add up to 640 (*i.e.* 20% of original nodes) new nodes

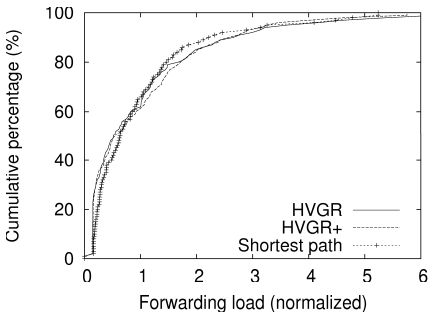

Fig. 10. Routing load balancing.

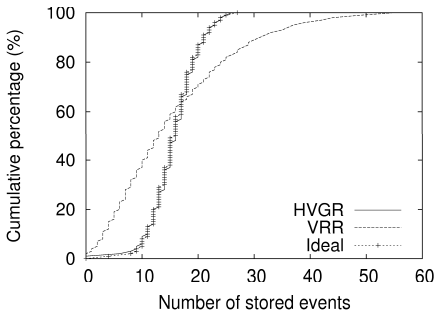

Fig. 11. Storage load balancing.

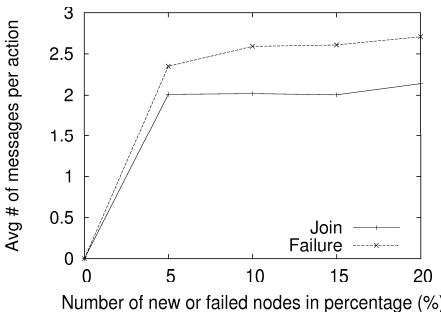

Fig. 12. Maintenance overhead of dynamic events.

randomly or remove up to 20% of nodes randomly. The join and failure frequency is about 1 event per second in average.

Figure 12 shows that the average extra control messages per join is very close to two. In most cases, the new node just sends one message to the lowest level landmark for join, and the landmark announces its join to other nodes in the lowest level region. Thus two control messages are needed. Occasionally, the new node is two hops away from the landmark and hence it becomes a new landmark. This case is rare, especially in dense networks.

Figure 12 also shows there is very small average control overhead when nodes fail silently. In average, a node failure causes less than three messages to repair the hierarchical virtual coordinate. As described in Section III-G3, in most cases a failed node does not cause any addition message. Our distance vector variant can heal quickly by leveraging on the piggy-backed sequence numbers in HELLO messages. A failed landmark is simply replaced by another node in the same region, and the new landmark announce the replacement by HELLO messages gradually.

V. Conclusions and Future Work

In this paper, we propose a new Hierarchical Voronoi Graph based Routing (HVGR) and its improvement HVGR+ for data-centric storage. Our HVGR is very scalable, as the initialization overhead and routing table size of each node is $O(\log N)$. HVGR and HVGR+ achieve close to shortest path performance as the region oriented routing and Hop ID based routing utilize the underlying network topology. We design a simple hash mechanism so that HVGR can provide a well load balanced data-centric storage system. In future, we plan to implement HVGR in real sensor testbed to further evaluate it in the realistic scenarios.

References

[1] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *ACM/IEEE International Conference on Mobile Computing and Networks (MobiCom)*, Boston, Massachusetts, Aug, 2000.

[2] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan, "Building efficient wireless sensor networks with low-level naming," in *8th ACM Symposium on Operating Systems Principles*, Oct, 2001.

[3] S. Shenker and et al, "Data-centric storage in sensornets," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 137–142, 2003.

[4] X. Li, Y. Kim, R. Govindan, and W. Hong, "Multi-dimensional range queries in sensor networks," in *the first international conference on Embedded networked sensor systems (Sensys)*, Nov. 2003.

[5] B. Greenstein et al, "DIFS: A distributed index for features in sensor networks," in *1st IEEE International Workshop on Sensor Network Protocols and Applications*, May, 2003.

[6] S. Ratnasamy et al., "GHT: A geographic hash table for datacentric storage in sensornets," in *1st ACM Workshop on wireless Sensor Networks ands Applications*, Sep., 2002.

[7] C.-T. Ee, S. Ratnasamy, and S. Shenker, "Practical data-centric storage," in *the Third USENIX/ACM NSDI,*, May, 2006.

[8] R. Fonseca et. al, "Beacon vector routing: Scalable point-to-point in wireless sensornets," in *2nd Symposium on Networked Systems Design & Implementation (NSDI)*, May, 2005.

[9] Q. Fang et al., "Glider: Gradient landmark-based distributed routing for sensor networks," in *IEEE Infocom*, Mar. 2005.

[10] J. Bruck, J. Gaoy, and A. Jiang, "Map: Medial axis based geometric routing in sensor networks," in *the 11th Annual International Conference on Mobile Computing and Networking(Mobicom)*, Aug. 2005.

[11] G. Pei, M. Gerla, X. Hong, and C.-C. Chiang, "A wireless hierarchical routing protocol with group mobility," in *IEEE Wireless Communications & Networking Conference (WCNC)*, 1999.

[12] Q. Cao and T. Abdelzaher, "A scalable logical coordinates framework for routing in wireless sensor networks," in *IEEE Realtime Systems Symposium*, Dec. 2004.

[13] A. Caruso, S. Chessa, S. De, and A. Urpi, "Gps free coordinate assignment and routing in wireless sensor networks," in *IEEE Infocom*, Mar. 2005.

[14] Y. Zhao et al., "Efficient hop id based routing for sparse ad hoc networks," in *IEEE International Conference on Network Protocols(ICNP)*, Nov. 2005.

[15] M. Caesar et. al, "Virtual ring routing: Network routing inspired by dhts," in *ACM SIGCOMM*, Sept., 2006.

[16] B. Karp and H. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in *the 6th Annual International Conference on Mobile Computing and Networking(Mobicom)*, Aug. 2000.

[17] M. Aly, K. Pruhs, and P. K. Chrysanthis, "KDDCS: A load-balanced in-network data-centric storage scheme in sensor network," in *ACM Conference on Information and Knowledge Management (CIKM)*, 2006.

[18] J. Newsome and D. Song, "Gem: Graph embedding for routing and data-centric storage in sensor networks without geographic information," in *the First International Conference on Embedded Networked Sensor Systems (SenSys)*, Nov. 2003.

[19] A. Rao, C. Papadimitriou, S. Shenker, and I. Stoica, "Geographic routing without location information," in *the 9th Annual International Conference on Mobile Computing and Networking(Mobicom)*, 2003.

[20] Y. Mao et. al, "S4: Small state and small stretch routing protocol for large wireless sensor networks," in *4th Symposium on Networked Systems Design & Implementation (NSDI)*, April, 2007.

[21] C. C. Chiang and M. Gerla, "Routing and multicast in multihop, mobile wireless networks," in *IEEE ICUPC*, Oct. 1997.

[22] G. Pei, M. Gerla, and X. Hong, "Lanmar: Landmark routing for large scale wireless ad hoc networks with group mobility," in *IEEE/ACM MobiHoc*, 2000.

[23] Z. J. Haas and M. R. Pearlman, "The performance of query control schemes for the zone routing protocol," *ACM/IEEE Transactions on Networking*, vol. 9, no. 4, pp. 427–438, Aug, 2001.

[24] P. Tsuchiya, "The landmark hierarchy: A new hierarchy for routing in very large networks," in *ACM SIGCOMM*, Aug., 1988.

[25] F. Aurenhammer, "Voronoi diagrams - a survey of a fundamental geometric data structure," *ACM Computing Surveys*, vol. 23, no. 3, pp. 345–405, 1991.

[26] A. S. Tanenbaum and M. van Steen, "Distributed systems: Principles and paradigms," *Prentice Hall*, Sept. 2001.

[27] C. Hedrick, "Routing information protocol," *RFC 1508*.

[28] F. Kuhn, R.r Wattenhofer, Y. Zhang, and A. Zollinger, "Geometric ad-hoc routing: Of theory and practice," in *Principles of Distibuted Computing*, 2003.