An In-Network Querying Framework for Wireless Sensor Networks

Murat Demirbas, *Member, IEEE,* Xuming Lu, *Student Member, IEEE,*
and Puneet Singla, *Member, IEEE*

***Abstract*— In contrast to traditional wireless sensor network (WSN) applications that perform only data collection and aggregation, new generation of information processing applications, such as pursuit-evasion games, tracking, evacuation, and disaster relief applications, require in-network information storage and querying. Due to the resource limitations of WSNs, it is challenging to implement in-network querying in a distributed, lightweight, resilient and energy-efficient manner. We address these challenges by exploiting location information and geometry of the network and propose an in-network querying framework, namely the Distributed Quad-Tree (DQT). DQT is distance sensitive for querying of an event: the cost of answering a query for an event is at most a constant factor ($2\sqrt{2}$ in our case) of the distance "d" to the event. DQT construction is local and does not require any communication. Moreover, due to its minimalist infrastructure and stateless nature, DQT shows graceful resilience to node failures and topology changes.**

Since event-based querying is inherently limited to the anticipated types of inquiries, we further extend our framework to achieve complex range-based querying. To this end, we use a multi-resolution algorithm, that is optimal with respect to least square errors that models the data in a decentralized way. Our model-based scheme answers queries with approximate values accompanied by certainty levels with increased resolution at lower layers of the DQT hierarchy. Our analysis and experiments show that our framework achieves distance-sensitivity and resiliency for event-based querying, as well as greatly reducing the cost of complex range querying.

***Index Terms*— Distributed Quad-Tree, Distance Sensitive In-network Querying, Multi-resolution modeling, Wireless Sensor Networks**

I. Introduction

WIRELESS sensor networks (WSNs) have been treated mostly as data collection and aggregation networks. Examples of such are WSNs deployed for environmental monitoring [32], [33] and military surveillance [1], [2]. As the WSN technology matured, instead of serving as passive information gathering mechanisms only, WSNs started to serve more as active information processing tools. Examples of these are pursuer-evader applications [8], smart building [6] etc., where mobile entities query the WSN on the spot to learn about their surroundings. Here latency and energy-efficiency become fundamentally important due to the real-time requirements of the tasks and the resource limitations of WSNs.

A major part of querying services is event querying. An event querying is used for checking whether a predefined event happened in a region. For instance, a soldier in a battlefield may need to know the location of the nearest enemy tank. Latency and energy-efficiency suffer drastically if these queries are always routed to basestations for resolution; therefore an in-network querying approach has been proposed and widely adopted in the literature [13], [36]. To be deployable in practice, certain requirements need to be satisfied by an in-network querying service. First of all, the in-network querying service needs to be distance-sensitive and also efficient for information storage. Distance-sensitivity for querying implies that the cost of answering a query for an event should be at most a constant factor "s" of the distance "d" to the event of interest in the network. Besides the distance sensitivity requirement, the in-network querying service should provide graceful resilience to the face of node failures. By graceful resilience, we mean that the performance degradation of querying should be commensurate with the severity of faults.

An event query is useful for checking whether an event happened in a region, however, event-based querying is inherently limited to the anticipated types of inquiries, and there is a need for range-based querying [18], [25], such as querying for an object that has similar features to those provided by the user. Range query is defined as a query that requests all the objects falling into a given range of interest. A range query can ask for joining, counting, MIN/MAX, arbitrary data, and statistics problems. Some examples of range querying are: find big red metallic objects, or list all events whose temperature is above $60F$ in an area. We call the first type "search" querying (includes joining, MIN/MAX etc) and the second type "lookup" querying (includes counting, statistics). In contrast to event-based querying, whose indexing reduces to denoting whether a specified event exists in that region or not, range-based querying poses a huge challenge for efficiently and properly indexing of arbitrary data. Indexing every sensor data (since the query can be about any arbitrary value) is out of question due to energy and storage constraints. To reduce the querying cost and yet also achieve efficient in-network indexing, modeling the sensor data is quite useful. In WSNs, sensed data from the environment is usually highly correlated both in spatial and temporal domains. For example, a node's temperature is highly correlated with nearby nodes, therefore, nearby nodes can be used for estimation with proper modeling. To be deployable in practice, the modeling needs to be performed in a decentralized, efficient, and lightweight manner.

Contributions: Our contributions for the in-network querying problem are two folds: (1) achieving distance sensitivity and resilience in event-based querying and (2) reducing the cost of range querying through modeling.

Event-based querying. We present an efficient and robust in-network querying infrastructure, namely Distributed Quad-Tree (DQT) [10], suitable for real-world WSN deployments. DQT overlays a quad-tree structure on a WSN and satisfies distance-sensitive in-network event querying. DQT is a hierarchical struc-

Murat Demirbas and Xuming Lu are with the Computer Science and Engineering Department, SUNY at Buffalo, Buffalo, NY 14260. E-mail: demirbas,xuminglu@cse.buffalo.edu

Puneet Singla is with the Department of Mechanical and Aerospace Engineering, SUNY at Buffalo, Buffalo, NY 14260. E-mail: psingla@buffalo.edu

ture and hence is suitable for multi-resolution information representation and querying. In contrast to extensive usage of quadtrees in a centralized manner [12], DQT is completely distributed. DQT maintains a minimalist structure, and in fact, DQT can be considered as stateless. DQT achieves this feat by employing an encoding technique that maps a quadtree over the deployment area by exploiting the location information. The implication is that the construction of DQT is local and does not require any message exchanges. The stateless operation of DQT makes it resilient to the face of node failures and topology changes. To achieve resiliency while routing to clusterheads or neighbors in the structure, DQT maps the DQT address of the destination to the physical coordinates, and leverages on the resilience of a geographic routing scheme (such as GPSR [22]) for delivering the message. GPSR re-routes the information to the closest node to the failure node, which we call the proxy node. The proxy node pretends to be the target node and finds its neighbors through local computation. Change of the shape of coverage holes only affects the selection of proxy nodes, and has little influence on other nodes.

Model-based range querying. To address the challenges of indexing arbitrary data, we present a model-based range querying framework, where the data is approximated by a set of weighted basis functions. The goal is to provide a lossy and progressive approximation model for answering range querying with minimum increased storage. One advantage of our model is that it does not require any prior knowledge of sensor data. To achieve a lightweight modeling, we use a novel optimal multi-resolution modeling algorithm. We apply the regression method to denote data correlations and reduce dimensionality at the bottom levels. We then take advantage of DQT hierarchy and perform global-local optimization at different levels to achieve an optimal multi-resolution modeling of the sensor data with least square errors, and store the resulting coefficients in the corresponding level clusterheads. Error covariance is propagated in the hierarchy and is used for quantifying on the accuracy at higher levels. Although the sensed data at the bottom layers may change, the higher layers are updated infrequently, only when the data changes are significant enough to trigger a modification of the higher layer modeling parameters. Thus, energy-efficiency of indexing is achieved in our framework. Our multi-resolution modeling is also useful in data collection/aggregation and distributed data exfiltrating to the basestation from WSN in an efficient manner.

To achieve usability despite the approximated values, we use the concept of approximation certainty based on error models. The certainty of a range query is defined as the probability of the estimated value Xi being in the range of $[a_i, b_i]$. The higher the probability of the approximation Xi being within $[a_i, b_i]$, the higher the certainty. If the result satisfies the user-defined certainty thresholds, it is deemed acceptable and the query is answered using models instead of diving deeply into the network to locate the physical nodes to gather the data. Besides energy-efficiency, another benefit of our model based querying is that the querying results do not need to rely absolutely on the live sensor readings, relaxing the requirement for very dense sensor coverage.

To validate our modeling and in-network querying algorithm, we analyze our model using real sensornets data. We compare the efficiency of model-based range querying with a naive range querying algorithm and show that our model-based algorithm greatly improves the querying efficiency and reduces querying costs.

Outline. We describe the DQT model and assumptions in Section 2. Encoding techniques and DQT construction are discussed in Section 3. Event querying is analyzed in Section 4 and our optimal multi-resolution model-based range querying is presented in Section 5. The simulation results serve as empirical validation of scalability, distance sensitivity, querying efficiency and resilience of DQT, and are presented in Section 6. We discuss related work at Section 7.

II. MODEL

We assume that the WSN motes sit on a two dimensional plane and their coordinates (x,y) are made available to themselves[1]. We assume a connected network and availability of geographic routing such as greedy perimeter stateless routing (GPSR [22]). There may exist some coverage holes in the network, but the network remains connected (i.e., no isolated regions). Our analytical results for event querying in DQT are proved in Section V in the absence of holes in the network, and in Section VII via simulations we show how they hold up in the presence of holes in the network.

As we describe in the next section, the network is divided into grid cells while embedding a DQT over the network. A level 1 box in DQT constitutes the smallest cell area in the DQT structure. We assert that all motes inside a level 1 box are within one hop distance, which put constraints on the least number of levels needed. In our terminology, a mote refers to a physical wireless sensor node, while a "node" refers to a virtual DQT node, such as a level 1 box. The cost of querying an event is measured as the number of hops traveled from the querying node to a node that holds an advertisement about the event. The cost of range querying is the overall number of hops traversed from the start querying node until results come back from all corresponding nodes.

III. DQT STRUCTURE AND CONSTRUCTION

For constructing DQT, we employ an encoding trick first described in [10]. Each partition divides a region into 4 sub-regions, which are encoded as 0,1,2,3 corresponding to NW, NE, SW and SE partitions. As such, each level 1 box in the structure is assigned an ID which uniquely identifies a region. The length of an ID is equal to the number of levels. We use this addressing scheme to preserve the location information of a node. Due to the way we construct level 1 nodes, this scheme is independent of the number of nodes (there may be multiple nodes in the same level 1 box), but relies on the partition levels. Fig.1 illustrates the addresses of the nodes in a partitioned region with 3 partition levels. Similar to the centralized quad-tree, DQT is a hierarchical structure. In each level of partition, a node is assigned as clusterhead of the corresponding region. The clusterhead at each partition is statically assigned to be the closest node to the geographic center of the entire region. For example, in level 1 partition, node 003 is selected as clusterhead for 00 region, because it is closer to center than nodes 000, 001 and 002. Similarly, node 033 is selected as level 2 clusterhead. Hence, the node closest to the center of the entire network in each sub-partition is selected as the parent node of that sub-partition. The benefit of such a selection is to avoid backward links. For instance,

[1]Our framework is easily extensible to 3-D space

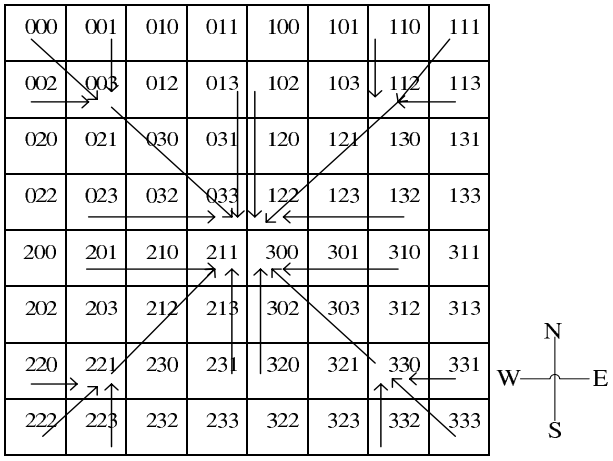


Fig. 1. Node addressing and hierarchical structure

in Fig.1, node 000 propagates the query to its root node 033 by first contacting parent node 003, then 003's parent 033. A DQT node may belong to different levels in the hierarchy depending on its location. If a node is a member at level k, it is also a member at all levels less than k. We denote a node p's parent as p.parent and children as p.child. The neighboring nodes are called siblings, which are denoted as p.sibling.

This structure is quite simple and adaptable to multidimensional sensor readings, such as (temp,light,humidity), since the construction of DQT does not rely on sensor values. The addresses of the clusterhead and neighboring clusterheads at each level for a given node are easily derivable arithmetically using the node's DQT address. By exploiting the location information DQT avoids a costly bottom-up construction in fact no extra communication cost is introduced for DQT construction as we describe next two subsections..

A. Mapping from localization to DQT addressing

Each node in DQT can calculate the DQT address of the level 1 partition it resides in from its x,y coordinates easily as we describe next. In our discussion, we let each level 1 partition to be a square($w = l$), even though slight difference between the width and length will not affect any results. The construction does not rely on the shape and node distribution, yet a uniform distribution will lead to better load balancing. For a WSN deployment as in Fig.2, we take the maximum bounding rectangle with two endpoints $(x_s^{'}, y_s^{'})$ at NW corner and $(x_e^{'}, y_e^{'})$ at SE corner. Then we calculate (x_s, y_s) and (x_e, y_e) denoting the corner of the DQT overlay as follows (if $|x_s^{'} - x_e^{'}|$ is larger than $|y_s^{'} - y_e^{'}|$):

$$y_s = y_s^{'} + \frac{|x_s^{'} - x_e^{'}| - |y_s^{'} - y_e^{'}|}{2}$$

$$y_e = y_e^{'} - \frac{|x_s^{'} - x_e^{'}| - |y_s^{'} - y_e^{'}|}{2}$$

and $x_s = x_s^{'}$, $x_e = x_e^{'}$. Basically, we transform a bounding rectangle to a bounding square by using the longer edge of the rectangle (Similar construction is used when $|x_s^{'} - x_e^{'}|$ is smaller than $|y_s^{'} - y_e^{'}|$). Consequently, the number of levels (i) of the DQT structure is:

$$i = log_2(\frac{|y_e - y_s|}{l})$$

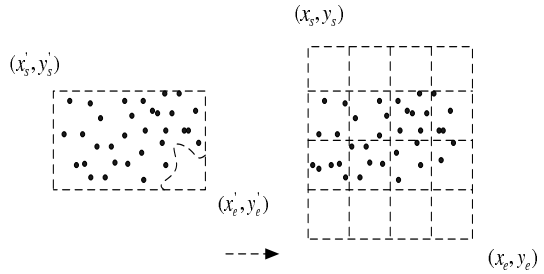


Fig. 2. Mapping from irregular shapes to squares

And the DQTID of a node(x,y) can be calculated as:

$$DQTID = \left[\left|\frac{x - x_s}{w}\right| (binary)\right] + \left[\left|\frac{y - y_s}{l}\right| (binary)\right] * 2$$

The mapping calculates the X and Y address separately, and then adds them together. Inside the bracket we use binary number system, while outside we consider these numbers as if decimal numbers. We can verify this formula from Fig.1. For instance, given node 033's location (assume the DQTID is not known), we get X address 011 and Y address 022 (011*2 in the equation). Hence the DQTID is 033 by adding 011 and 022. The reason that the second term in the DQT address calculation is multiplied by 2 is because Y addresses pace by 2 for every increment in DQT addressing scheme. Given this mapping, any node can locally compute its DQT address based on its coordinates (x,y).

Besides the DQT address, each node also maintains its (x,y) coordinate address. This location information is used in GPSR message routing for querying and advertising. Since GPSR only requires single hop information, which has already been cached as level one neighbors in our structure, it is very suitable for WSNs. When the coverage has irregular holes, local optimal path can be reached using right hand rules in GPSR. By adopting the above encoding trick and assigning DQT addresses for DQT nodes, we next discuss the construction of the DQT structure.

B. DQT Local Construction

DQT uses local construction instead of bottom-up construction to reduce communication cost during initial construction. A static and local scheme that uses the address of the box suffices for calculating every level clusterheads and neighbors at N, S, E, W, NE, NW, SE and SW. In the following we discuss how to find the clusterhead and neighbors.

The clusterhead validate algorithm provides the relation of DQT address to its clusterhead. We find that in NW region of the map, nodes with DQT-address "3" at level i and lower positions (denoted as p.address(i*)) become the clusterheads at the corresponding level. Similarly, in NorthEast partition, nodes with DQT-address "2" at level i become the clusterheads. In Fig.3, p.address(h) is the highest bit of the DQT-address, which determines the region of a node. This algorithm guarantees the clusterheads at each level are closer to the map center than any children (except for itself).

To find the neighbors, we again make use of the location information. We use node p and node q to represent the originator and its neighbor. First we use p's location information and increase its coordinates x, y value by a level i box lateral length to find a neighbor node in each direction. If either of x or y value exceeds the range of the map, we ignore that neighbor. For each of

```
Procedure Cluster_head_Validate (node p, level i)
  Switch (p.address(h))
  Case 3: //p in SE region
  { If p.address(i*) == 0, then return true; else return false}
        Case 2: /p in SW region
  { If p.address(i*) == 1,then return true, else return false}
        Case 1: //p in NE region
  { If p.address(i*) == 2, then return true, else return false}
        Case 0: // p in NW region
  { If p.address(i*) == 3, then return true, else return false}
```

Fig. 3. Clusterhead validate algorithm

these nodes, we find their level i clusterheads. These clusterheads are node p's level i neighbors. For instance, given a node 20l at level 2, we can find its neighbor at north direction by following two steps: (1) Reduce Y value by a level two box length, then we can locate the node 021 through the new (x,y) coordinates; (2). Find the clusterhead for node 021 at level 2, which is 023.

IV. EVENT QUERYING IN DQT

Before discussing querying in detail, we show how events are indexed in DQT.

A. Indexing of event information

000 #	001 #	010 #	011	100	
002 #	003	012 #	013 ##	102	
020 #	021 #	030 #	031	120	
022 #	023 ##	032 #	033 ##	122	
200	201	210	211	300	

Fig. 4. Node 003's indexing structure

In any hierarchical structure as with DQT, some multilevel boundary nodes are far away from each other in the structure, while actually they are placed nearby in the network. High latency maybe introduced if the search follows the path of the tree structure strictly. For example in Fig.1, node 011 and 100 are neighbors. A query from node 011 to node 100 may route to higher level clusterheads such as node 013 and node 033. Our solution is to use sibling links to nearby intermediate nodes. A sibling link is the link between a node and its neighbors in each direction (so each may at most have 8 sibling neighbors). The sibling links only exist between nodes on the same level in the structure. Fig.4 illustrates the point of view of an intermediate node 003 in DQT structure. The nodes with "#" are level 1 sibling nodes, the nodes with "##" are level 2 sibling nodes. A node at level i maintains the event information of its cluster, as well as the event information of its neighbors. When an event is detected at a level 1 node p, p contacts its immediate parent node at level 1. The parent node updates its record for that child. Node p also contacts its sibling nodes to update their records accordingly. Recursively, the update operation is executed till the top level. This is similar to the information storage scheme discussed in [13] and the sibling links in Stalk [9].

B. Event Querying

Our discussion of event query includes, but is not limited to Nearest Neighbor(NN) query (NN query is defined as, finding the data object which is closest to the querying object given a set of objects). In WSNs, a query can be started at any location. The initiator of a query is the node where the query is entered into the system. The query point is the node for which we want to get the result. Query point by default is the same node as initiator of query but it may be specified to be any point in the network.

Our algorithm prevents the propagation of searching to higher levels if it can be answered locally. Through taking advantage of the spatiality information, both the query efficiency and latency are greatly improved. Our query strategy is to start the query at the query point using local information because the node may belong to multiple levels and therefore hold multi-layer information locally. If no result is obtained, the query is propagated to the parent recursively. In this way, the querying message remains atomic without making multiple copies into the network. At some level the event information is reached, the query is then stopped and returned to the originator.

What if the query point is at another location? That means the initiator of the query and the query point belongs to two different nodes. First the query is passed to the query point from the initiator of the query using GPSR routing scheme, and then this querying process is started from the query point. The following results are in the absence of faults. In the simulation section, however, the results are achieved in the presence of faults.

Lemma 1. *DQT spends $O(i)$ space to store an event upto level i.*

Proof. An event is indexed at most in 9 nodes (including the detecting node) at each level. Hence, the space needed for indexing an event upto level i is at most $O(i)$.

Theorem 1. *The total space needed for DQT is less than $O(nh)$, where n is the total number of level 1 nodes and h is the height of the DQT structure.*

Proof: According to Lemma 1, level 1 nodes use up $9*n$. Since any node at level i has at most 4^i children, level i nodes need $\frac{9n}{4^i} * 4^i$ in all. Thus the total space needed for DQT is:

$$\sum_{i=1}^{h}(\frac{9n}{4^i} * 4^i) = 18nh = O(nh)$$

Lemma 2. *The distance between a level i and its neighbors is at most $2^i * \sqrt{2}$ hops*[2]

Proof: According to the partition rule of quad-tree, a level i node is the clusterhead of a $2^i * 2^i$ area. The distance between a level i node and its neighbors is either 2^i (for N, S, E, W neighbors) or $2^i * \sqrt{2}$ (for NE, NW, SE, SW neighbors) depending on the direction. Since the clusterhead is one of its neighbors at level i, so the distance between a level i node and its clusterhead is also less than $2^i * \sqrt{2}$ hops, which is the diagonal distance of a level i partition.

Lemma 3. *The overhead of an event reporting in DQT is $O(D)$ in a uniform distribution, where D is the diameter of the field.*

Proof: According to Lemma 2, the distance from level i-1 node to its parent node (level i) is $2^{i-1} * \sqrt{2}$ hops. Thus the overhead

[2]The result is based on the assumption that width equals to length for each level 1 box

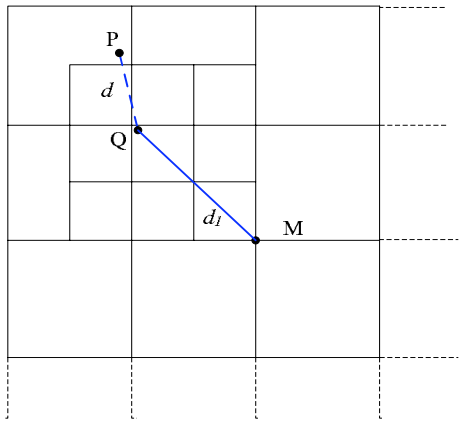


Fig. 5. Distance stretch factor s analysis

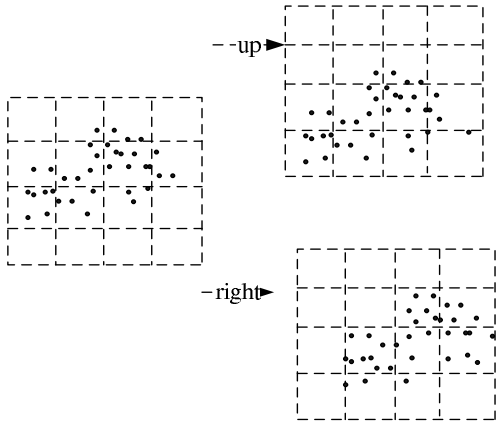


Fig. 6. Shifting the square of DQT field helps to balance the load.

of an event reporting is less than:

$$\sum_{i=1}^{\log_4 n} 8\sqrt{2}(2^{i-1}) = O(\sqrt{n})$$

In case, sensor nodes are uniformly distributed, $\sqrt{n} = D$. Hence the overhead of an event reporting in DQT is $O(D)$.

Theorem 2. *The distance stretch factor s for spatial query in DQT is $2\sqrt{2}$ in worst case, when the query point is the same node as the query initiator. In another words, an event d hops away can be achieved by the querying node within $d * 2\sqrt{2}$ hops.*

Proof: A query from an intermediate level node does not constitute the worst case. The reason is that the clusterhead nodes holds multi-levels information locally and this local cache can be used to answer queries. So, lets consider a query from a bottom level node that reaches a level j clusterhead. We define the query cost as the number of hops from the query point to the node that holds the result.

In Fig.5, d_1 is the distance from querying node Q to highest level of node M that the query is propagated; d is the distance from Q to P, where P is the destination node that node Q is querying. Distance stretch factor s is defined as $s = d_1/d$.

According Lemma 2, the distance from level i-1 node to its parent node (level i) is $2^{i-1} * \sqrt{2}$ hops. Since the backward links are avoided in going-up phase, the total distance from level 1 to level j can be calculated as $(1+2^1+2^2+\cdots+2^{j-1})*\sqrt{2}$, which is overall $d_1 = \sqrt{2} * (2^j - 1)$ hops. Since P and Q are not i-1 level neighbors, the distance $d \geq 2^{j-1}$. The equivalence is true when P and Q are located exactly on the opposite borders of a level i-1 box. Hence:

$$s = d_1/d \leq \sqrt{2} * (2^j - 1)/2^{j-1} < 2\sqrt{2}$$

Based on the result on case 1 and case 2 analysis, we conclude that our structure is distance sensitive with a distance stretch factor $2\sqrt{2}$.

C. Fault tolerance

DQT is fault tolerant due to several reasons. First, any leaf node failure is masked without causing any update operation and structure change. This is because, for a dense sensor network, each level 1 partition contains several nodes and all nodes in the same partition share a common DQT ID. Moreover, since DQT structure is stateless, the nodes do not need to maintain a state of its own, they act on behalf of other nodes in the level 1 box.

Second, DQT can handle coverage holes nicely. Only if all the motes inside a level 1 partition fail, a hole may be formed in DQT. In the case of failures of motes in an area, GPSR delivers a message addressed to a box in that area to a mote on the boundary of the hole. Since DQT is stateless, the recipient mote easily acts as a proxy on behalf of the intended destination ID, and determines the next step in the query or advertisement operation by simply plugging the destination ID into the corresponding procedures for the DQT operation. (Proxy checks the destination_ID and realizes the message is not intended for itself, then proxy uses destination_ID to calculate the next steps required for routing the message). This way, failures of motes in an area degrade the performance of DQT operations proportional to the size of the area. Essentially, the degradation is equal to that of routing stretch in GPSR due to the holes. DQT preserves correct functionality unless the network is partitioned, and even then, functionality is satisfied within each partition.

D. Load balancing

Static hierarchical configuration and clusterhead election lead to unbalanced energy consumption at various levels. High level nodes are more frequently utilized and prone to depleting. Load balancing can be achieved by shifting the square periodically, such that the sensor motes acting as clusterheads can be rotated in some extent.

Fig.6 shows how shifting the square can be used to improve load balancing. In practice, the network can be programmed to shift the field perhiodically in DQT construction, e.g., up-down shifting and left-right shifting. Through this manner, clusterheads at all levels are alternated among their neighboring nodes. DQT pays no extra costs for such alternation since the DQT is maintained as stateless.

V. RANGE QUERYING

In this section we discuss range querying algorithms for DQT. We compare two types of range querying algorithms: naive range querying and model based range querying.

A. Naive Range Querying

We describe a naive algorithm to solve the range querying problem. The straightforward strategy to resolve range queries

is only to make use of the geometric information. Suppose the query range is enclosed within points (x_1, y_1) and (x_2, y_2). First we calculate the span of each direction S_h and S_v:

$$S_h = log_2(\left|\frac{x_2 - x_1}{w}\right|) + 1$$

$$S_v = log_2(\left|\frac{y_2 - y_1}{l}\right|) + 1$$

where w and l are the width and length of level 1 box. Then we can easily see that horizontal constraint is within two level-S_h regions, and vertical constraint is within two level-S_v regions. Overall, the constrained area lies within two level MAX (S_h, S_v) neighbors. Thus the query is directly propagated to a level MAX (S_h, S_v) node covering that area. The algorithm is shown in Fig.7. The idea is to find the least level node which can cover range constraints and then send the query to that node using GPSR protocol. The reason for using a top-down brute force search algorithm to answer a range query in Fig.7 is that we cannot prune any nodes inside the specified area without having some information about the values of those nodes. In naive range querying the network does not keep any information about the sensor values in a region, and this is exactly the problem that the model based range querying is trying to address.

```
NaiveRangeQuery (Node q, Nodeset p) {
        Calculate i = Max(Sh,Sv);
        Route the query to corresponding level i node q';
        Start breadth-first-search from q';}
```

Fig. 7. Algorithm for naive range query

B. Model based range querying

In this section we first present the range querying algorithm for modeled DQT data. Then, in the following subsections, we discuss how to efficiently model sensor data and approximate data with certainty and error bounds. Similar to the naive range querying, in the model-based approach the query is also forwarded to the least level clusterhead that covers the range constraints. However, unlike naive range querying, lower level nodes are not contacted to acquire data unless the model is insufficient. The difference between model based range querying algorithm and naive range querying algorithm is the if statement in Fig.8, which is used to determine whether or not to send queries to lower levels. For this purpose, we use certainty associated with approximation variation as stopping criteria. For a given atomic query problem (complex range query is composed by atomic queries), the certainty $C(X_i - \epsilon, X_i + \epsilon)$ is calculated as probability of $P(X_i \in [X_i - \epsilon, X_i + \epsilon])$. A query whose certainty level is already satisfied by the modeled data stops exploring further, hence the communication cost and energy consumption are reduced.

C. Sensor data modeling

An important aspect in model based query is the accurate mathematical representation of the physical field measured by the WSN since a high fidelity model of the measured physical field is generally not available as priori and has to be constructed in real-time. Let us assume that we have m data collections

```
ModelBasedRangeQuery (Node q, Nodeset p) {
Calculate i = Max(Sh,Sv);
Forward the query to corresponding level i node q';
Calculate the confidence with error bound;
if (C(Xi-ε, Xi+ε)>= threshold)
    Return estimated results;
else
    Send the query to children;
}
```

Fig. 8. Answering the range querying with modeled data

$(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)$. For the simplest mathematical representation, the mapping from the input variable x to the measurable variable y can be approximated by a linear algebraic equation of the following form:

$$y_i \approx a_0 h_0(x_i) + a_1 h_1(x_i) + ... + a_n h_n(x_i),\ i = 1, 2, \cdots, m \quad (1)$$

So the problem of finding the appropriate mathematical model will reduce to the estimation of unknown parameters (a_i) from certain data measurements. When the approximation implicit in Eq. (1) is satisfactory, we have a linear algebraic estimation problem. Since $n < m$, the coefficients vector compressed the original data by $\frac{n}{m}$ ratio. An estimated value of unknown coefficients $\boldsymbol{alpha} = \{a_0, a_1, \cdots, a_m\}^T$ can be found by minimizing the the vertical offset between measured values and estimated values.

$$\hat{\boldsymbol{\alpha}} = (\mathbf{H}^T \mathbf{H}^{-1}) \mathbf{H}^T \mathbf{Y} \quad (2)$$

where $\mathbf{Y} \in \mathcal{R}^m$ is a vector of measured values of signal $f(x_i)$ at m distinct points and $H_{ij} = h_i(x_j)$ is an $m \times n$ matrix of independent basis functions. Note that relationship of Eq. 1 represents just the approximation of the actual (true) input-output process and is suspectable to any modeling and sensor errors.

The process of globally approximating an unknown physical process by means of least-square methods provides a clean and optimal solution given a set of observation. However, collecting every child's data is expensive and the computation is too complex for distributed sensor nodes, especially for large networks. Instead, since the regression usually includes a specified tolerance of error, it is possible to represent DQT data in an efficient manner so that the approximation accuracy toward some certain interests such as Root Mean Square (RMS) error meets the specified criterion at a certain level. A key question regarding the proper selection of a mathematical model of a physical process is "How irregular is the variations of the model over space and time?" A global best fit as described above should be sufficient if the variations in physical process are smooth globally. In the presence of localized distortions, a more judicious selection of the mapping approach is required. Furthermore, many advanced approximation methods such as Artificial Neural Networks, Wavelets etc. provide both a qualitative and a theoretical motivation for using a linear combination of a large number of nonlinear functions to approximate irregular phenomena but in actuality, they offer no guarantee on accuracy in practice for a reasonable dimensionality. In the next section, we briefly discuss a multi-resolution approximation framework to approximate the unknown physical process while considering the aforementioned issues, i.e., computational and communication cost constraints. The main goal of the proposed approach is to reduce computational complexities by creating

a multi-resolution structure and then processing the query data at different resolutions. We utilize coarse and fine representations to capture global and local characteristics respectively. This decomposition is natural and is feasible because the global characteristics are expected to be in the lower end of the spatial frequency spectrum and the local characteristics are expected to be in the upper end of the spatial frequency spectrum.

D. Global-local multi-resolution algorithm

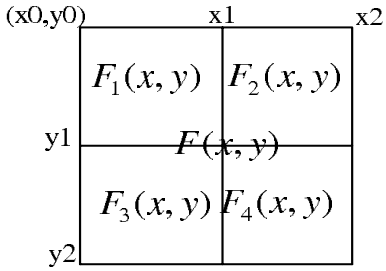


Fig. 9. Global-Local multi-resolution process

Multi-resolution approximation is an iterative process hierarchically decomposing the input-output approximation and is an important property for any approximation algorithm. Spline approximations [4], Wavelets [7] and Finite Element Methods (FEM), are most commonly used multi-resolution algorithms. Although global-local separation of concern leads to immensely improved approximation algorithms; the main drawback of conventional multi-resolution algorithms is that one can not use different basis functions to obtain different local approximation without introducing discontinuity across the boundary of different local regions for a specific multi-resolution algorithm. For example, in the case of wavelet based approximation, one should use same wavelet function at different resolution levels and similarly, in case of B-spline, one is restricted to use only polynomial basis functions in various intervals. Ideally one may like to choose these basis functions, based on prior knowledge about the problem or based solely upon local approximability using Sparse approximation methods. While such freedom provides can greatly improve the approximability, it generally prevents the basis functions from constituting a conforming space; i.e., the inter-element continuity of the approximation is not ensured. Hence, there is a need of rigorous methods to merge different independent local approximations to obtain a desired order globally continuous approximation and this is the main feature of the proposed multi-resolution algorithm. At the heart of our approach is a recently developed powerful method for blending independently derived local approximations into consistent global approximations [29]–[31].

In DQT, each level 1 cluster-head performs regression based on its direct children's values. High level models only give general ideas of the distribution of sensor values. At lower levels, the model becomes more and more detailed and precise and thus, provides better approximation of the actual physical process. We denote the node space at the lowest level as Ω, the node space at second level Ω^2, the third level Ω^4, $\cdots$ and so on. A power of two is used in the node space labeling because each higher level potentially covers 4 times as many nodes as the previous level.

In the DQT hierarchy, a parent node is divided into four quadrants that are completely decoupled from each other as shown in Fig. 9. Let us assume that $F_1(x,y)$, $F_2(x,y)$, $F_3(x,y)$, and, $F_4(x,y)$ are four independent local models for each of the four quadrants at the granularity level Ω^1. These preliminary local approximations $F_i(x,y)$ are completely arbitrary, as long as they are smooth and represent the local behavior of unknown physical process well. Since these local models are completely arbitrary, there is no guarantee of inter-quadrant continuity. Although the discontinuities at the quadrant boundaries do not affect any event querying and range querying of discrete nodes, they lead to estimation ambiguities at the quadrant boundaries which is not desirable. We address the inter-element continuity problems directly using recently developed means for blending independently derived local approximations into consistent global approximations [29], [31]. Given the local approximations $F_i(x,y)$ and special weighting function $w(x,y)$, the weighted average approximation is defined as:

$$F(x,y) = \sum_{i=1}^{4} w_i(x,y) F_i(x,y) \tag{3}$$

The weighting function $w(x,y)$ is used to blend or average the four adjacent preliminary local approximations $F_i(x,y)$, $i = 1,2,3,4$ and a generic expression for the weighting functions can be obtained by imposing the following boundary value problem, as discussed in detail in [29], [31]:

1) The first derivative of the weighting function must have an d^{th}-order osculation with $w(0,0) = 1$ at the centroid of its respective local approximation.
2) The weighting function must have an $(d+1)^{th}$-order zero at the centroid of its neighboring local approximations.
3) The sum of all neighboring weighting functions must be unity over the entire closed interval between their corresponding adjacent local functional approximations.

If weighting function is assumed to be polynomial in an independent variable, then adopting the procedure listed in [29], [30] the weight function for first order continuity can be shown to be simply:

$$w(x,y) = \ (1-\bar{x}^2(3-2\bar{x}))(1-\bar{y}^2(3-2\bar{y}))$$

In [29], generalized expressions for these weight functions have been developed for desired order of continuity d. A fundamental theoretical result [29] thus obtained states that if the local approximations are un-biased estimations of the input-output data, then: (i) the final blended approximation is un-biased, (ii) the variance of the blended approximation is substantially smaller than the variance of any of the averaged approximation, (iii) the mathematical structure of the averaged approximation can be varied as appropriate to optimally capture the local geometry, and finally, (iv) the blending weight functions guarantee the global piecewise continuous nature of the approximation.

Finally, to find the expression for a coarser model while minimizing the approximation error on the global model at Ω^2, we define the following cost function:

$$J = \frac{1}{2}\int_{y0}^{y2}\int_{x0}^{x2}\sum_{i=1}^{4}[F(x,y) - w_i F_i(x,y)]^2 dxdy \tag{4}$$

where w_I is the special averaging function associated with i^{th} quadrant and the global model at Ω^2 granularity is assumed to be

a linear combination of l pre-defined basis functions:

$$F(x,y) = \sum_{j=0}^{l} b_j \phi_j(x,y)$$

where l is restricted to be less than the number of coefficients required to represent local approximation at finer level, i.e., Ω^1. Further, if basis functions h_j are chosen to be orthogonal to each other, then the coefficients b_j can be computed efficiently.

E. Sequential Processing

There are many engineering application problems which needs to be solved in an iterative manner in real-time by successively approximating the input-output data. It is desirable that cluster-heads update their models upon receipt of new corresponding data subset. Linear sequential regression technique such as Kalman filter [5] can be used to handle this problem.

Let us assume that $\mathbf{Y}_{k-1}$ is the past observed data that follows the assumed measurement model $\mathbf{Y}_{k-1} = \mathbf{H}_{k-1}\boldsymbol{\alpha}_{k-1}$, and $\mathbf{Y}_k$ is the new observed data, then we can use the following Kalman update equations for computing the new estimation of $\boldsymbol{\alpha}_k$:

$$\boldsymbol{\alpha}_k = [\mathbf{I} - \mathbf{K}_k\mathbf{H}_k]\boldsymbol{\alpha}_{k-1} + \mathbf{K}_k\mathbf{Y}_k$$

$$\mathbf{K}_k = \mathbf{P}_k\mathbf{H}_k^T$$

$$\mathbf{P}_k^{-1} = \mathbf{P}_{k-1}^{-1} + \mathbf{H}_k^T\mathbf{H}_k$$

$\mathbf{K}_k$ is the Kalman gain matrix and $\mathbf{P}$ is the state covariance matrix. This method can be applied recursively to make model corrections in real time. This also hints to a good way of incorporating temporal information into our framework to estimate past, current, and future trends. The temporal model may use either distinct sample at each interval or spatial models at each interval. The former method requires more space since each node keeps some history samples while the latter one only stores model information. A learning phase can be trained to estimate the best sampling intervals. If the temporal model is known or even partially known, it can be useful in reducing the modeling complexity or making sampling interval more adaptive. For example, if it is highly likely that during $t \in [t_1, t_2]$, temperature happens to change drastically, the models can be updated more frequently to reflect real time fluctuations. In [34], a theoretical result is given between approximation error and updating frequency.

F. Stopping criteria

We presented the adaptive refinement process in above section without discussing the stopping criteria. Suppose at level i the stopping criteria is η_i, then we have $\eta_1 \geq \eta_2 \geq \cdots \geq \eta_l$, where l represents the maximum number of levels.If η_i already satisfies the query, the querying process stops at level i.

If the sample is small, then we assume $T = (\epsilon - \mu)/(\sigma/\sqrt{n})$ follows t-distribution. T-distribution is a special case of normal distribution. When the sample size is large, it becomes Gaussian distribution. The probability density function (PDF) of t-distribution is already known [23]. If the certainty is 90% and the value T is within the interval [-A, A], we say with 90% certainty:

$$-A < \frac{\epsilon - \mu}{\sigma/\sqrt{n}} < A$$

$$\Leftrightarrow \mu - \frac{A\sigma}{\sqrt{n}} < \epsilon < \mu + \frac{A\sigma}{\sqrt{n}}$$

For example, given a sample of 9 elements with variance equals 1 and mean 5, we can determine that at 90% certainty (A= 1.397) the interval is:

$$[5 - 1.397\frac{1}{\sqrt{9}}, 5 + 1.397\frac{1}{\sqrt{9}}] = [4.53, 5.47]$$

Finally, we are able to define our stopping criteria as the certainty with certain approximation variation. If the approximation certainty level with certain variation is satisfied, the querying process terminates.

G. Further Discussion

With multimedia capability being more and more available at the sensor nodes, WSNs are becoming capable of performing object classification, image segmentation, motion analysis etc. There has been some work in Wireless Multimedia Sensor Networks (WMSNs) such as Cyclops [27] and multimedia QoS routing [22]. Our model provides a general and application independent framework to perform multimedia in-network processing. To avoid transmitting large amounts of raw data to the sink, the multimedia raw data can be filtered and extracted of semantically useful information and stored in multi-dimensional vectors. Many feature extraction techniques, such as color histogram and gray scale [19], vector approximation [16], shape indexing [3] etc., can be used to represent and index images. If images are represented in high dimensional vectors, dimension reduction techniques (e.g PCA) can be applied to reduce the dimensionality. Again we may use regression based techniques to model the multi-dimension scalar data at various levels by a set of weighted basis functions. This also provides a high efficiency distributed compression and multi-resolution fusion scheme, since multimedia data is highly correlated. In such a framework, multimedia queries can be evaluated at different levels and only when demanded the raw image data will be transmitted to the initiator. This greatly reduces the transmission of redundant information and increases the lifetime of the system.

VI. SIMULATIONS AND SAMPLE DATA ANALYSIS

We investigate the performance characteristic of DQT using the ns-2 wireless network simulator. Our simulations and numerical analysis mainly focus on event querying and range querying. The settings for event querying simulation are as follows: 256 nodes are uniformly distributed in a 2-dimensional square of $3200m \times 3200m$. The distance between each node is 200m, and the transmission range is set to be 250m. Therefore the average degree is about 4 in the field except some border nodes. That is, not all the level 1 neighbors are reachable via single hop. The geographic location of each node is available, and is used to construct the DQT structure in initial phase. The height of the DQT tree is 4, with 4 roots at the top level. The cost of querying an event is measured as the number of hops from the querying node to the node that holds an advertisement about the event.

A. Stretch factor in event querying

We have proved in Theorem 2 that the stretch factor in worst case is $2\sqrt{2}$. We calculate the average distance stretch factor through 100 runs of each experiment. In each round, a query/sink

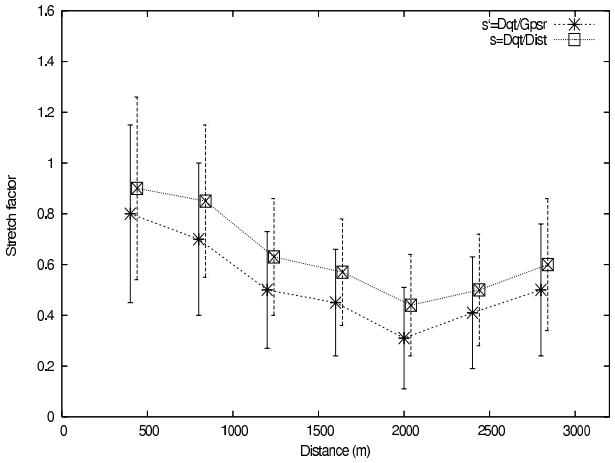


Fig. 10. Stretch factor s and s' with varied query distance

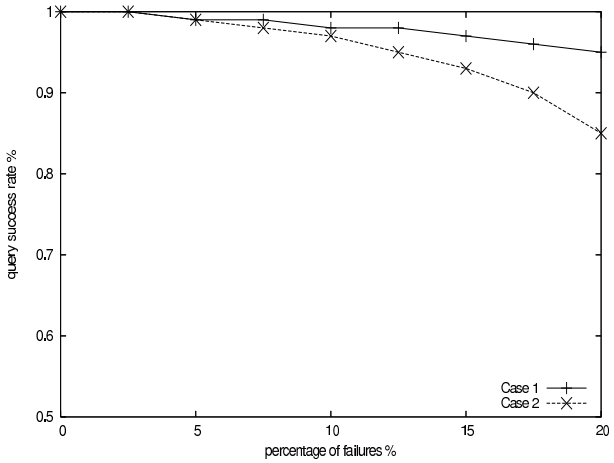


Fig. 11. Query success rate for case 1 and case 2

node pair is randomly chosen. We use two measurements s and s', where s is the ratio of the DQT querying cost to the hops between the query and the event node and s' is the ratio of the DQT querying cost to the GPSR routing cost. The value of s shows the ratio to ideal cost, whereas s' shows the ratio of routing a message from the querying point to the event. We found the average s is around 0.6 and s' is around 0.5 in the absence of faults. The reason that s is much smaller than $2\sqrt{2}$ is that the worst case scenarios only occupy a small percentage of all cases. Our scheme has a considerably smaller stretch factor compared to the DSIB scheme, which has an average value 0.9~1. As an event has been indexed in the structure, it is not surprising that the average stretch factor can be even much less than 1.

Fig.10 illustrates the average stretch factor s and s', as well as their standard deviations where the event and querying pairs are randomly selected with varied distance between the query node and event node. For nearby pairs, the s and s' is close to 1, since either GPSR or DQT makes little difference. The average stretch factor s decreases with the increase of the distance of query/event pairs. But we also find, that when the distance is close to the diameter of the map (such as 2800m or 3200m), s and s' slightly increase again. We call this phenomenon border effect. The reason is that when the pair of nodes approach the borders of the maps, they are less likely to be connected through their common neighbors.

B. Fault tolerance

A single mote failure will not change the DQT structure and the query operation. To evaluate the performance of DQT, node failure in the structure is simulated. Failures may cause the following two cases in event querying. Case 1: Failures happen before the event advertisement. When a target node of event advertisement fails, the event is published to proxy node by default, which is the closest node to the failure node. The failure of advertising destination node will not affect the query result in theory, since it can reach the proxy node. Case 2: The event has already been published in the structure before the failure happens. When a node with event advertisement dies, queries to this node are passed to its parent node. In this case, the event is still reachable unless all the nodes along the querying path were dead. We will further discuss the performance of each case in simulation part.

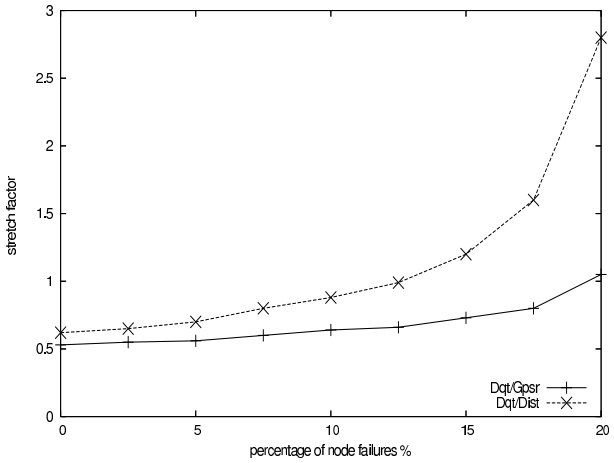


Fig. 12. Stretch factor with node failure: Case 1

We keep the same topology and setting in our experiment as before. Our first experiment on the DQT is the query success rate with node failure. We randomly remove a certain percentage of nodes, say from 2% to 20%. Note that, for case 1, the event still publishes in proxy node when the destination node fails. Theoretically there is no failure for querying, unless GPSR fails to forward along the path or the network is isolated. In case 2, the query is extended to the parent node when a node with event advertisement fails. A query may fail when all event nodes along the querying path fail. Fig.11 shows that for case 1, query success rate can drop down to 95% when 20% of nodes fail. However, for case2, the DQT scheme itself may fail besides the GPSR routing failure. The failure rate goes up to 15% in case 2. The result may vary in real environment due to the increase of GPSR failure and link asymmetry. Increasing the degree of nodes or node density is helpful in improving the query success rate.

From the stretch factor point of view, case 2 is also worse than case 1. Fig.12 and Fig.13 illustrate the stretch factor with varied possibilities of node failure for case 1 and case 2. In both cases, DQT works fairly well within 10% failure of nodes. With the increase in the failure rate, the stretch factor s gets worse quickly for both cases, because the query circumvents the holes (due to the failure), which increases the cost of searching rapidly. Case 1 performs better than case 2 because in the occurrence of holes,

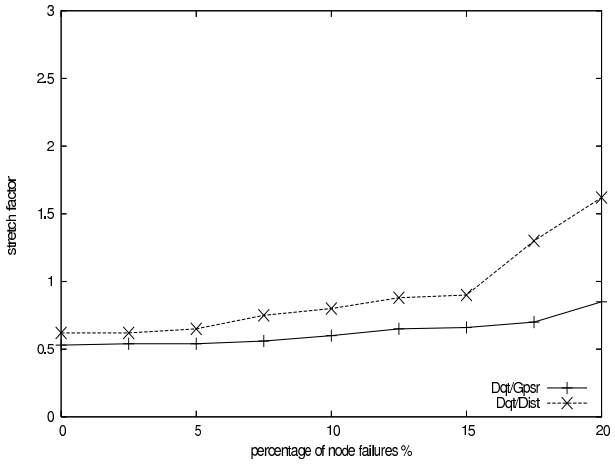


Fig. 13. Stretch factor with node failure: Case 2

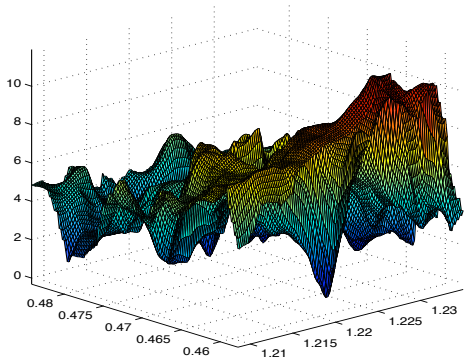
case 2 circumvents the hole and query its parent; while in case 1, the event is achievable through a proxy node. The s' remains relatively small since the same overhead applies for GPSR to overcome the coverage holes. The results also indicate that the degradation of performance is smooth overall.

C. Sample Data Analysis for range querying

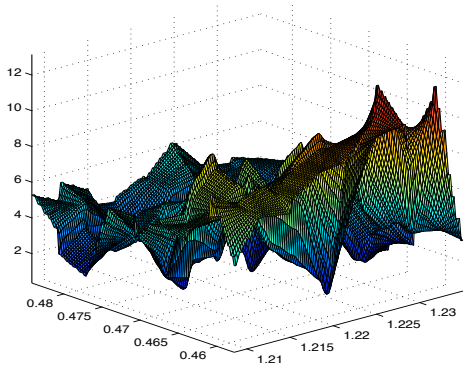
To verify the range querying algorithms in modeled DQT, we use sample dataset from PRISM group [35]. PRISM is an analytical model that uses point data and a digital elevation model (DEM) to generate grid estimates of monthly and annual average daily maximum/ minimum temperatures. The resolution of this sample data is 0.0417decimal degrees both in Latitude and Longitude, which is about 4km. The map total covers a region 256x256 km2. Although the dataset is at a significantly larger scale than usual sensor networks, the data exhibits spatial-temporal correlations, providing a useful case study in testing our algorithm.

We first form a 64x64 grid dataset using average Max temperature set in January 2006. We assume these 64x64 grid data form bottom level framework. As stated in the previous sections, we know that at level 1 there are 32x32 nodes, level 2 has 16x16 nodes. We start regression process at level 2 nodes (thus each regression contains 16 data observations). Every higher level model is derived of its local models using global-local multi-resolution algorithm. We analyze the data in second order spatial model. The approximation results are illustrated in Fig.14 at various levels. From these figures, we expect that at lower levels, the approximation is very close to real data.

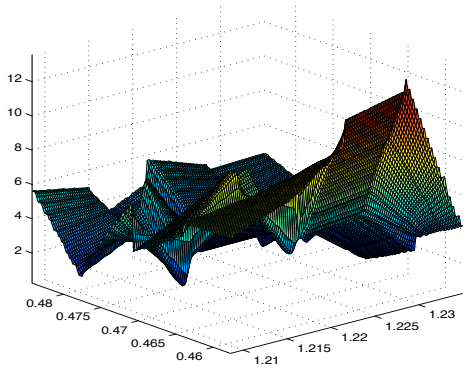
Another lightweight implementation of multi-resolution model is to carry out regression process (instead of global-local algorithm) at each level by mapping node space from low level to high levels, which we call average mapping based distributed regression. Of course, various mapping methods could be applied; a simple way is to calculate the average value in this region and take it as the value at the center point, which is stored at clusterheads. We compare the performance of global-local algorithm with this average mapping scheme in Fig.15 with 80%, 90% and 95% certainty levels. The approximation error accumulated quickly at high levels. It shows that at bottom levels (such as level 2 and level 3), the variances are close for both schemes; however, at high levels, variances in average mapping scheme accumulates



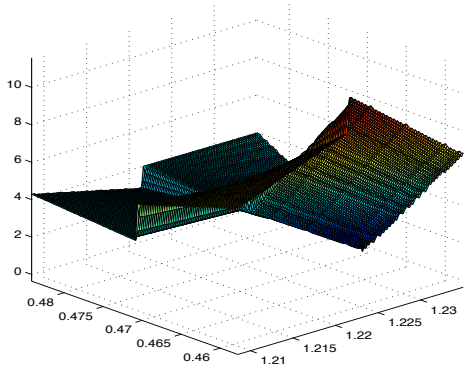
(a) Level 2



(b) level 3



(c) Level 4



(d) Level 5

Fig. 14. Hierarchical multi-resolution plots at level 2,3,4,5 respectively

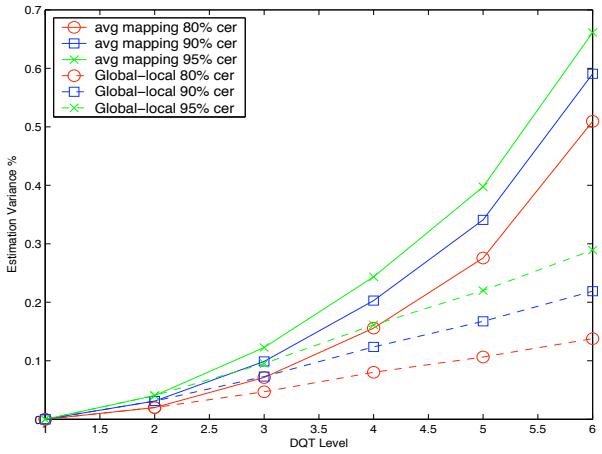


Fig. 15. Mean approximation variance(global-local algorithm)

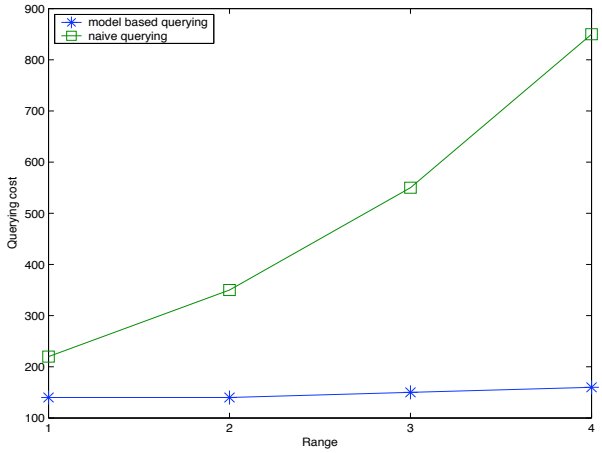


Fig. 16. Lookup querying cost comparison

more sharply than global-local algorithm. The reason is that average-mapping only considers the mapped nodes and loses the information of nearby nodes. The benefit is that average mapping scheme requires less computation and storage overheads. At level 2, we can expect 3% variance with 90% certainty in average of all nodes in both schemes, while at top level the average variance is accumulated up to 25% for global-local mapping algorithm and 60% for average mapping algorithm in our sample. If we allow 10% of approximation variance with 90% certainty, level 3 would be enough for answering queries using global-local algorithm. For each specific level, higher certainty leads to larger approximation variance. There are two possible ways to handle an unsatisfied query: either by reducing the certainty requirement or by sending the query to lower layers until the requirement is satisfied.

The cost of a range query depends on each query and it varies greatly for different instances. We measure the cost as the overall number of hops for a query from being initiated until being answered. We give some concrete examples for two types of range querying: lookup querying and searching. All queries assume 10% approximation variance with 90% certainty. Fig.16 considers the following lookup querying instances. Instance 1a: Find the temperatures at nodes 01320 and 23100. Instance 2a: Find the Instance 2a: Find the temperatures at space ranges [Lon (122.10,122.15): Lat (46.35,46.99)] and [Lon (123.18,123.23): Lat (47.77,47.78)](Note: the range contains node 01320 and 23100). Instance 3a is the same query with doubly sized range in instance 2a and instance 4a doubly sized range in instance 3a. Fig.17 considers following search querying instances. Instance 1b: Find the nodes whose temperatures are greater than 5.0 degree in the range [Lon (122.10, 122.15): Lat (46.35, 46.41)]. Instance 2b, 3b, and 4b are the same querying with doubly sized range to each prior. We compare the cost of model-based querying (as in Fig.8) with naive querying scheme (as in Fig.7) by assuming the query originator's ID 00000.

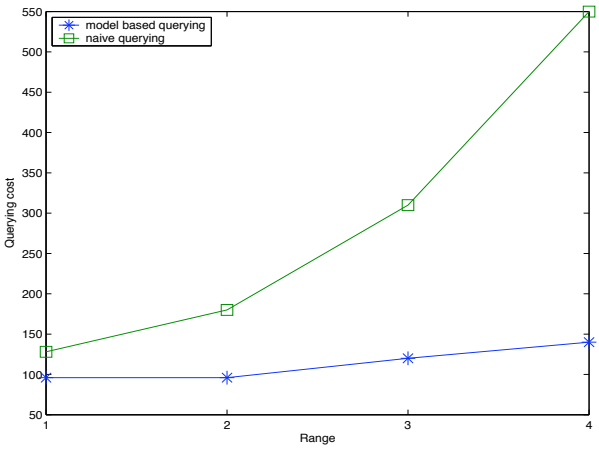


Fig. 17. Searching querying cost comparison

We can see that model based querying strategy outperforms naive querying algorithms in all scenarios. Model based querying is efficient for lookup querying problems since it is not obligated to send the query to every node in the range. For searching problems, model based approach can filter nodes at high levels in favor of models, thus performs more efficiently than naive querying especially when the range constrains are not very tight. For both types of querying problems, model based querying is not as sensitive as naive querying to the change of range constraints. The reason is that model based approach incorporates spatial correlation in approximation while naive querying needs to explore every node in the range.

VII. RELATED WORK

In this section, we study the related work on WSNs and compare the differences with our proposed framework.

Storage and querying: Centralized querying has been the common mode of querying in WSN. For this mode of operation, the basestation acts as the point where the query is introduced and results are gathered. For example, in TinyDB [26], queries are first parsed at the basestation and disseminated into the WSN to be executed. This centralized structure is not feasible for distributed and self-organizing sensor networks since: (1) such a basestation may not exist, (2) for in-network queries, a query may be introduced from any node in the network and (3) propagating the query to the base station can be costly.

Geographic Hash Tables (GHT) [28] gives a simple solution for in-network event querying problem: GHT stores and retrieves information by using a geographic hash function on the type of the event. GHT can hash event information far away from the nearby query nodes, and thus violate the distance sensitivity of querying. The average cost of storing and querying in GHT is D/3, where

D is the diameter of the network. In general, information storage and querying efficiency are inversely related pairs in optimization. For example, directed diffusion [21] chooses to optimize the information storage (O(1) cost) to the extent of querying ($O(d^2)$ cost). Combs&needles optimizes querying $O(1)$, to the extent of information storage $O(d^2)$, or can achieve the vice versa.

Distance Sensitive Information Brokerage (DSIB) protocol [13] achieves distance-sensitivity in a hierarchically partitioned network by using a push-based approach: an event advertises to neighbors as well as its parents at every level of the hierarchy. DSIB does not require localization information and relies purely on communication topology. To this end, DSIB introduces a costly bottom-up construction and a special purpose routing algorithm. In contrast to DSIB, DQT assumes localization information and in turn is able to provide an efficient local construction. Our another work Trail [24] achieves distance sensitive object tracking, however, it only supports tracking-based applications and is not capable of searching arbitrary data as well as range querying. As a summary, we compare the performance of DQT with that of other querying solutions in Table I.

TABLE I
INSERTION AND QUERYING PERFORMANCE COMPARISON.

	Insertion	Query
Flooding	$O(N)$	$O(1)$
Diffusion	$O(\sqrt{N})$	$O(\sqrt{N})$
GHT	$O(\sqrt{N})$	$O(\sqrt{rN})$
DSIB	$O(D)$	$O(d)$
Trail	$14dlogd$	$38d$
DQT	$8\sqrt{2}D$	$2\sqrt{2}d$
D-Diameter; N-Number of nodes; d = distance from query point; r-Number of discrete values		

Data modeling: In many systems [35], [37], WSNs have been studied from a database point of view, which has no way to handle approximation queries. For instance, a querying toward a location without a sensor simply returns no result. There is no ambiguity of possible error associated with results. Toward approximated query processing, substantial work have been done in the AQUA [17] and CONTROL [20] projects. AQUA is based on distinct sampling scheme to provide approximation results for distinct value queries. AQUA is centralized and is not based on a spatial algorithm, and hence is unable to exploit the correlations of neighboring nodes. CONTROL provides an interactive method for handling long running complex queries; a query usually starts with a broad, big-picture querying and then is continually refined based on feedback with certain certainty bounds. These centralized schemes do not help much in distributed sensor network environment without proper aggregation algorithms.

Our model-based range querying algorithm is partly inspired by BBQ [11], a WSN query processing framework that incorporates statistic analysis on available sensor data both in spatial and temporal domain. BBQ uses a time varying multivariate Gaussian model and answers queries with probabilistic certainty. BBQ provides a practical algorithm of optimizing querying execution plan by selecting the best sensor readings to acquire and balance the certainty as well as the communication and data acquisition costs in the network. However, BBQ lacks the capability to handle in-network querying, since such a complex optimization problem needs to be handled by a basestation. Another constraint of BBQ is that it assumes that the probabilistic distribution function (PDF) is known. In contrast to BBQ, DQT enables in-network querying and does not assume prior knowledge of sensor data.

DIMENSIONS [14], [15] provides a unified view of data processing in WSNs, handling data storage, multi-resolution data access and spatial-temporal pattern mining. In its wavelet model, compressed data from lower levels are first decompressed at higher level clusterheads, and recompressed with the jointed and larger dataset, which requires higher computing capability and complexity on hardware devices. Instead, we use a regression based model, in which data can be constructed directly without any wavelet encoding and decoding processes as in [15]. In addition, our model needs constant extra space at each level, storing the coefficients of basis functions. Model-based DQT also differs from distributed kernel regression algorithm [19] in that in kernel regression model, kernels are represented by a normalized kernel weight function with predefined kernel regions. Kernel regression is only effective for familiar and reachable environments, and is not applicable for an unpredictable area.

VIII. CONCLUDING REMARKS

We presented an in-network querying infrastructure, namely distributed quad-tree (DQT) structure, suitable for use in real world WSN deployments. DQT satisfies distance-sensitive event querying as well as efficient information storage in network. DQT construction is local and does not require any communication. Moreover, due to its minimalist infrastructure and stateless nature, DQT shows graceful resilience to node failures and topology changes. In fact, it is possible to extend DQT to provide a location service for mobile ad hoc networks. The idea is to retry a query until it catches up with the mobile target. Even though a target node may move during the query execution and leads to a miss, the query when invoked from this new location closer to the target node will have a better chance to catch up to the target node due to the distance-sensitivity property in DQT.

Furthermore, to enable ad hoc on-the-fly querying of users, we extended our framework to deal with the complex range queries and the associate challenge of indexing arbitrary data in the network. We presented a model based framework for answering querying applications on a hierarchical DQT network. The global-local multi-resolution algorithm used in our modeling process is optimal in terms of Root Mean Square propagation errors. In our framework, each node only holds the coefficients of approximating functions, saving the cost of storage and achieving an efficient way of data compression.

Our work has not touched the querying optimization problem with multiple attributes, for which more correlations may get involved. Our current work is on extending our framework to address this problem in the context of similarity querying of image sensors in WSNs. Finally, we will investigate applications of our multi-resolution modeling framework in the distributed boundary/shape detection and distributed change detection domains.

REFERENCES

[1] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, and H. Zhang et al. A line in the sand: A wireless sensor network for target detection, classification, and tracking. *Computer Networks (Elsevier)*, 46(5):605–634, 2004.

[2] A. Arora, R. Ramnath, and E. Ertin et. al. Exscal: Elements of an extreme scale wireless sensor network. *11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, 2005.

[3] J. S. Beis and D.G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces,. In *CVPR*, page 1000, 1997.
[4] C. D. Boor. *A Practical Guide to Splines*. Springer, 1978.
[5] J. L. Crassidis and J. L. Junkins. *Optimal Estimation of Dynamic Systems*. Chapman & Hall/CRC, 2004.
[6] S. K. Das, D. J. Cook, A. Bhattacharya, E. Heierman, and J. Lin. The role of prediction algorithms in the mavhome smart home architecture. *IEEE Wireless Communications (Special Issue on smart home)*, 9(6), 2002.
[7] I. Daubechies. *Ten Lectures on Wavelets*. Number 61 in CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial & Applied Math, Philadelphia, PA, 1 edition, 1992.
[8] M. Demirbas, A. Arora, and M. Gouda. Pursuer-evader tracking in sensor networks. *To appear in Sensor Network Operations, IEEE Press*, 2006.
[9] M. Demirbas, A. Arora, T. Nolte, and N. Lynch. A hierarchy-based fault-local stabilizing algorithm for tracking in sensor networks. *8th International Conference on Principles of Distributed Systems (OPODIS)*, pages 299–315, 2004.
[10] M. Demirbas and X.Lu. Distributed quad-tree for spatial querying in wireless sensor networks. In *Proc. IEEE International Conference on Communication (ICC)*, June,2007.
[11] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *In Proc. of VLDB*, 2004.
[12] R. Finkel and J.L. Bentley. Quad trees: A data structure for retrieval on composite keys. acta informatica. In *Acta Informatica*, pages 1–9, 2007.
[13] S. Funke, L. J. Guibas, A. Nguyen, and Y. Wang. Distance sensitive routing and information brokerage in sensor networks. In *In Proceedings DCOSS*, 2006.
[14] D. Ganesan, D. Estrin, and J. Heidemann. Dimensions: Why do we need a new data handling architecture for sensor networks? In *In Proceedings of the ACM Workshop on Hot Topics in Networks*, pages 143–148, 2002.
[15] D. Ganesan, B. Greenstein, D. Perelyubskiy, D. Estrin, and J. Heidemann. An evaluation of multi-resolution storage for sensor networks. In *In Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*, pages 89–102, 2003.
[16] A. Gersho. Vector quantization and signal compression. In *Kluwer Academic Publishers*, 1992.
[17] P. B. Gibbons. Distinct sampling for highly-accurate answers to distinct values queries and event reports. In *In Proc. of VLDB*, 2001.
[18] B. Greenstein, D. Estrin, R. Govindan, S. Ratnasamy, and S. Shenker. Difs: A distributed index for features in sensor networks. *First IEEE Ineternational Workshop on Sensor Network Protocols and Applications*, May 2003.
[19] C. Guestrin, P. Bodik, R. Thibaux, M. Paskin, and S. Madden. Distributed regression: an efficient framework for modeling sensor network data. In *In Proceedings of IPSN*, 2004.
[20] J. M. Hellerstein, R. Avnur, A. Chou, C. Hidber, C. Olston, V. Raman, T. Roth, and P. J. Haas. Interactive data analysis with control. *IEEE Computer*, 32(8):51–59, 1999.
[21] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Transsaction on Networking*, 11(1):2–16, 2003.
[22] B. Karp and H. T. Kung. Gpsr: greedy perimeter stateless routing for wireless networks. In *MobiCom: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 243–254, 2000.
[23] S. Kotz and S. Nadarajah. Multivariate t distributions and their applications. In *Cambridge University Press*, 2004.
[24] V. Kulathumani, A. Arora, M. Demirbas, and M. Sridharan. Trail: A distance sensitive network protocol for distributed object tracking. In *European conference on Wireless Sensor Networks (EWSN)*, 2007.
[25] X. Li, Y. J. Kim, R. Govindan, and W. Hong. Multi-dimensional range queries in sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems, Los Angeles, California, USA*, pages 148–159, 2003.
[26] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. The dedign of acquisitional query processor for sensor networks. In *ACM SIGMOD, Int. Conf. on Management of Data*, June 2003.
[27] M. Rahimi, R. Baer, O. Iroezi, J. Garcia, J. Warrior, D. Estrin, and M. Srivastava. Cyclops: in situ image sensing and interpretation in wireless sensor networks. In *In SenSys*, 2005.
[28] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. Ght: a geographic hash table for data-centric storage. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 78–87, 2002.
[29] P. Singla. Multi-resoultion methods for high fidelity modeling and control allocation in large scale dynamical systems. In *Texas A&M University*, 2005.
[30] P. Singla and J. L. Junkins. Global local orthogonal mapping (glo-map) in n-dimensions: Applications to i/o approximation. In *6th Conference on Dynamics and Control of Systems and Structures in Space*, 2004.
[31] P. Singla and J. L. Junkins. Multi-resolution methods for modeling and control of dynamical systems. In *CRC Press*, 2008, In Press.
[32] R. Szewczyk, A. Mainwaring, J. Polastre, and D. Culler. An analysis of a large scale habitat monitoring application. In *Sensys*, 2004.
[33] G. Tolle, J. Polastre, R. Szewczyk, N. Turner, K. Tu, P. Buonadonna, S. Burgess, D. Gay, W. Hong, T. Dawson, and D. Culler. A macroscope in the redwoods. *In Proceedings of the Third ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2005.
[34] M. C. Vuran, O. B. Akan, and I .F. Akyildiz. Spatio-temporal correlation: Theory and applications for wireless sensor networks. *Elsevier Computer Networks (COMNET) Journal*, 45(3):245–259, 2004.
[35] M. Widmann and C. Bretherton. 50 km resolution daily precipitation for the pacific northwest.
[36] Y. Yao and J. Gehrke. The cougar approach to in-network query processing in sensor networks. *ACM SIGMOD Record*, 31(3):9–18, 2002.
[37] Y. Yao and J. E. Gehrke. Query processing in sensor networks. *Proceedings of the First Biennial Conference on Innovative Data Systems Research (CIDR 2003)*, 2003.