An In-Network Querying Framework for Wireless Sensor Networks

Murat Demirbas, *Member, IEEE*, Xuming Lu, *Student Member, IEEE*
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 of interest in the 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.

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 regression algorithm, which is optimal with respect to least square errors and models the data in a decentralized way. Our model-based scheme answers queries with approximate values accompanied by confidence levels with increasing resolution confidence 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

1 Introduction

Wireless sensor networks (WSNs) have been treated mostly as data collection and aggregation networks. Examples of such are WSNs deployed for environmental monitoring [21][25] 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 [7], smart building [20] 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 WSN.

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 battle-field 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 [11]. 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 for querying 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. Beside 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 (e.g., is there a fire?). However event-based querying is inherently limited to the anticipated types of inquiries, and there is a need for complex range-based querying, such as querying for an object that has similar features to those provided by the user. Some examples of range querying are: to find big red metallic objects or to report synopsis of sensor values in an area. We call the first type "search" querying and the second type "lookup" querying. In contrast to event-based querying, whose indexing reduces to denoting whether a specified event exists in that region or not, complex 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

- *Murat Demirbas and Xuming Lu are with the Computer Science and Engineering Department, University at Buffalo, Buffalo, NY 14260. E-mail: {demirbas,xuminglu}@cse.buffalo.edu*
- *Puneet Singla is with the Department of Mechanical and Aerosapce Engineering, University at Buffalo, Buffalo, NY 14260. E-mail: psingla@buffalo.edu*

Manuscript received (insert date of submission if desired).

correlated with nearby nodes, therefore, nearby nodes can be used for estimation with proper modeling.

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 complex range querying.

Event-based querying. We present an efficient and robust in-network querying infrastructure, namely Distributed Quad-Tree (DQT), 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 structure and hence is suitable for multiresoltuion information representation and querying. In contrast to extensive usage of quad-trees in centralized manner [10], DQT is completely distruted.

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 quad-tree over the deployment area by exploiting the location information. The implication is that the construction of DQT is local and does not require bottom up construction at all. 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 [18]) for delivering the message. GPSR re-routes the information to the closest node to the target 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 and complex range querying, we present a model-based range querying framework using the DQT structure, where the data is approximated by a set of weighted basis functions. In contrast to some work in modeling sensor data, our model does not require any prior knowledge of data distribution.

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 confidence based on error models. The confidence of a range query is defined as the probability of the estimated value Xi being in the range of [ai, bi]. The higher the probability of the approximation Xi being within [ai, bi], the higher the confidence. If the result satisfies the user-defined confidence 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, which are likely to get lost due to faults.

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

2 Related Work

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[21], 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 is costly.

Geographic Hash Tables (GHT) [23] 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 [16] 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 [11] 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.

In many systems [30][32][21], WSNs have been studied from a database point of view. These centralized systems do not suffice for in-network querying. Moreover, WSNs treated as purely a database, 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 [3] and CONTROL [17] 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 confidence 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 [6], 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 confidence. BBQ provides a practical algorithm of optimizing querying execution plan by selecting the best sensor readings to acquire and balance the confidence as well as the communication and data acquisition costs in the network. However, BBQ lacks of 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 data model.

DIMENSIONS [14] provides a unified view of data processing in WSNs, handling data storage, multi-resolution data access and spatial-temporal pattern mining. In its implementation, multi-resolution data extraction is achieved by engaging every node in data filtering and wavelet compression, which requires higher 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 [14]. Model-based DQT also differs from distributed kernel regression algorithm [13] 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.

3 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[16]). There may exist some coverage holes in the network, but no partitions (i.e., 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 assume that all motes inside a level 1 box are within one hop distance. In our terminology, a mote refers to a physical WSN node, while a "node" refers to a virtual DQT node, such 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.

4 DQT Structure and Construction

For constructing DQT, we employ an encoding trick [9]. 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, 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

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

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.

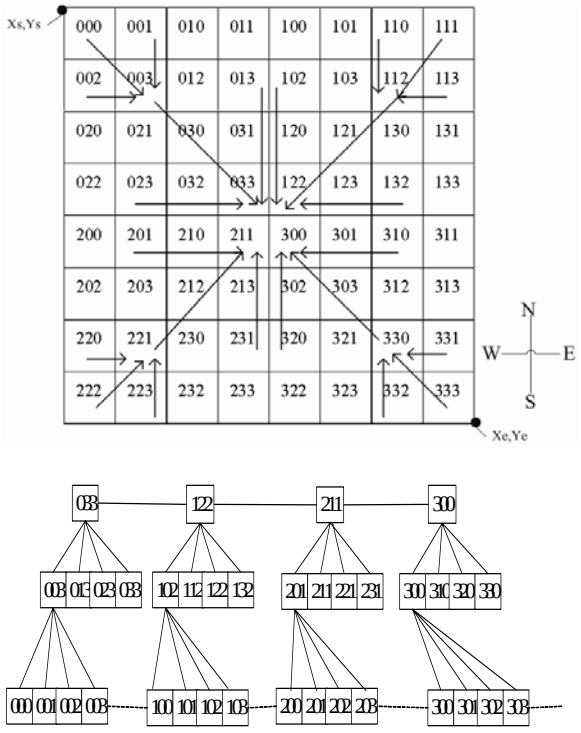

Figure 1: Node addressing and tree structure

This structure is quite simple and adaptable to multi-dimensional 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 thus no extra communication cost is introduced.

4.1 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. Let (x_s,y_s) at NW and (x_e,y_e) at SE be the two endpoints of the area where DQT should be overlaid. Assume DQT have i levels. The area of each level 1 box of partition is $w*l$ where width and length are:

$$l=(y_e-y_s)/2^i$$

$$w=(x_e-x_s)/2^i$$

Then DQT address of a node(x,y) can be calculated as:

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

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 quite simple to adapt to WSN. 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 can start constructing the DQT structure.

4.2 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. Each node may have 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.2, 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).

```
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}
```

Figure 2: clusterhead validate algorithm

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 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.

5 EVENT QUERYING IN DQT

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

5.1 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	

Figure 3: node 003's indexing structure

In any hierarchical structure as with DQT, some multi-level 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.3 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 it 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 [11] and the sibling links in Stalk [8].

5.2 Nearest Neighbor Query

Nearest Neighbor (NN) query is defined as, finding the data object which is closest to the querying object given a set of objects. The classic NN query returns exactly one object as a result. 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 NN query 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 is 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. 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, we talk about the results in the presence of faults.

Lemma 1. *A DQT node at level i stores O (i) information.*

Proof. A node at level i is clusterhead from level 1 to i along the path. The number of neighbor nodes at each level is less than or equal to eight. Therefore the node needs 9*i (including one record for its subtree) space and stores O (i) information.

Theorem 1. *The total space needed for the construction of DQT is less than 12*b, where b is the total number of level 1 nodes.*

Proof: According to Lemma 1, level 1 nodes use up 9*b space. Similarly, all level 2 nodes total to a 9*b/4 space usage. Thus, the total space needed for constructing the distributed quad-tree is:

$$9 * (b + \frac{b}{4} + \frac{b}{4^2} + ... + 1) = 12 * b(1 - \frac{1}{b}) < 12\, b$$

Lemma 2. *The distance between a level i node 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.

Theorem 2. *The distance stretch factor s for spatial query in DQT is* $2\sqrt{2}$ *in worst case. 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, let's 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.

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

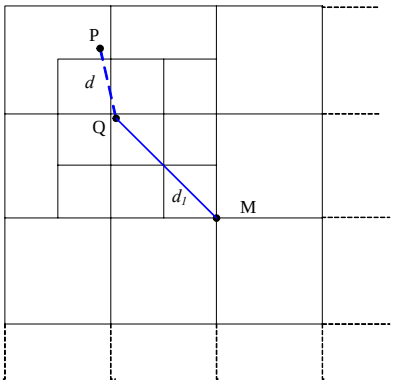

Figure 4: Distance stretch factor s analysis

In Fig.4, 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 $(1+2^1+2^2+\ldots+2^{j-1}) * \sqrt{2}$, 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 analyses, we conclude that our structure is distance sensitive with a distance stretch factor $2\sqrt{2}$.

5.3 Fault tolerance

DQT is fault tolerant due to several aspects. 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. 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.

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 decomposition direction periodically, such that the clusterheads can be rotated at various regions. Instead of choosing the clusterhead toward the center, they can be easily modified to any other directions. In those cases root nodes will be distributed to each geographic corners. DQT pays no extra costs for such alternation since the DQT is maintained as stateless.

6 Range Querying

In this section we discuss range querying algorithms for DQT. Two types of range querying algorithms are compared: naïve range querying and model based range querying. Naïve range querying is simple and easy for implementation especially when no additional knowledge of sensor data is available. We then propose a model-based range querying solution to improve querying efficiency and reduce cost. We use regression based techniques to model the scalar data at various levels by using a set of weighted basis functions.

6.1 Naive Range Querying

In this section, we describe a naive algorithm to solve the range querying problem. The straightforward strategy to resolve range queries is to make use of the geometric information. Suppose the query range is enclosed within points (x1, y1) and (x2, y2). First we calculate the span of each direction S_h and S_v:

$$S_h = \log_2 \left(\left| \frac{x2 - x1}{l} \right| \right) + 1$$

$$S_v = \log_2 \left(\left| \frac{y2 - y1}{w} \right| \right) + 1$$

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. 5. 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. 5 is that we cannot prune any nodes inside the specified area without having some information about the values of those nodes. In naïve 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 bread-first-search from q';}
```

Figure 5: Algorithm for naive range queries.

6.2 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 confidence 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 naïve 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. 6, which is used to determine whether or not to send queries to lower levels. For this purpose, we use confidence associated with approximation variation as stopping criteria. For a given atomic query problem (complex range query is composed by atomic queries), the confidence $C(X_i-\varepsilon, X_i+\varepsilon)$ is calculated as probability of $P(X_i \in [X_i-\varepsilon, X_i+\varepsilon])$. A query whose confidence level is already satisfied by the modeled data stops exploring further, hence the communication cost and energy consumption are reduced.

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

Figure 6: Answering range querying with modeled data.

6.3 Sensor data modeling

An important aspect in model based query is the mathematical representation of the physical field measured by the WSN since a high fidelity model of the measured physical field is generally not available a priori and has to be constructed in real-time. Suppose we have n data collections $(x_1, x_2, x_3 \ldots, x_n)$, we can fit the values in a m degree polynomial function: $f(x)=\alpha_0+\alpha_1 x+\alpha_2 x^2+\ldots+\alpha_m x^m$ (m<n for regression). The basis functions here are $(1, x, x^2, \ldots, x^m)$. More generally, this can be replaced by any basis functions as $(h_1, h_2, \ldots, h_n)$. In order to fit these values in the polynomial equation, we need to know m coefficients $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$. Since m < n, the coefficients vector compressed the original data by n/m ratio. We express the model using the function matrix **X**, target vector **Y** and parameter vector $\boldsymbol{\alpha}$. Considering regression error ε, the equation can be expressed as:

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\alpha} + \boldsymbol{\varepsilon} \quad (1)$$

where

$$\mathbf{Y} = \begin{pmatrix} f(x_1) \\ f(x_2) \\ \ldots \\ f(x_n) \end{pmatrix}, \mathbf{X} = \begin{pmatrix} h_1, h_2, \ldots, h_n \\ h_1, h_2, \ldots, h_n \\ \ldots \\ h_1, h_2, \ldots, h_n \end{pmatrix}, \boldsymbol{\alpha} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \ldots \\ \alpha_k^n \end{pmatrix}$$

The *ith* row of **Y** represents the regression value at the *ith* data sample (x_i, y_i).The equation can be solved using least square approximation which minimizes the mean squared error. The approximation function is described by the relation:

$$f(x) \approx \hat{f}(x) = \sum_{j=1}^{m} h_j(x)\alpha_j$$

The co-efficients α is obtained with minimization of residual error. Solving (1), the coefficients vector is achieved:

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

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 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. In next section, we briefly discuss our multi-resolution approximation framework to intuitively represent the physical process. The goal of the multi-resolution 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. At the heart of our approach is a powerful and novel means for blending independently derived local approximations into consistent global approximations[5] [26][27]. Furthermore, the approximation framework is amenable to recursive learning, whereby new measurements can be efficiently utilized to dynamically update the mathematical model.

6.4 Global-local multi-resolution algorithm

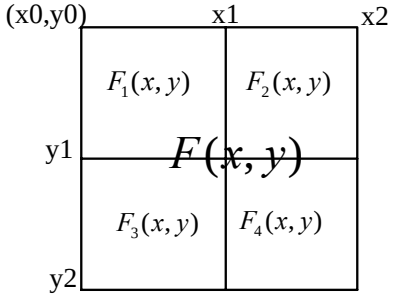

Figure 7: Global-local multi-resolution process

The above method provides a clean and optimal solution of approximation 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 speci-

fied criterion at a certain level. A multi-resolution based algorithm maybe needed to determine a global best fit based on local models. DQT is a hierarchical structure, and this simplifies the process by limiting the number of local models. The error estimation and stopping criteria are analyzed in the next section.

Multi-resolution approximation is an iterative process hierarchically decomposing the input-output approximation. In DQT, each level 1 clusterhead 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 gives better approximation. We denote the node space at the lowest level as Ω, the node space at second level Ω^2, the third level Ω^4,…, 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.

The bottom-up multi-resolution process maps a model in 2^mx2^m space to 2^{m-1}x2^{m-1} space as shown in Fig. 7. Suppose in Ω^1, $F_1(x,y)$, $F_2(x,y)$, $F_3(x,y)$, $F_4(x,y)$, are approximation models at Ω for each of the four quadrants. The goal is to minimize the approximation error on the global model at Ω^2:

$$J = \frac{1}{2}\int_{y0}^{y2}\int_{x0}^{x2}\sum_{l=1}^{4}\left[F(x,y) - w_l F_l(x,y)\right]^2 dxdy \qquad (3)$$

where

$$w_1 = \begin{cases} 1 & if\ (x0 \le x \le x1\ \&\ y0 \le y \le y1) \\ 0 & otherwise \end{cases}$$

$$w_2 = \begin{cases} 1 & if\ (x1 \le x \le x2\ \&\ y0 \le y \le y1) \\ 0 & otherwise \end{cases}$$

$$w_3 = \begin{cases} 1 & if\ (x0 \le x \le x1\ \&\ y1 \le y \le y0) \\ 0 & otherwise \end{cases}$$

$$w_4 = \begin{cases} 1 & if\ (x1 \le x \le x2\ \&\ y1 \le y \le y2) \\ 0 & otherwise \end{cases}$$

We consider the case where $F_l(x,y)$ is the nth order of linear combination of predefined local basis functions $\{h_j(x,y)\}$, e.g.,

$$F_l(x,y) = \sum_{j=0}^{n} a_{ij} h_j(x,y)$$

To simplify the model, we consider the global model with the same order of basis functions, e.g.,

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

To find the optimal solution of coefficient vector $\vec{\mathbf{b}}$ in global model, we can take derivative of J with respect to each parameter and set them equal to zero. Thus for k = 0,…n, we have:

$$\sum_{j=0}^{n} < h_j, h_k > b_j = < \sum_{i=1}^{4} \omega_i \sum_{j=1}^{n} a_{ij} h_j, h_k >$$

where < , > stands for the double integration for x and y. It can further be written as:

$\mathbf{H} * \vec{\mathbf{b}} = \vec{\mathbf{f}}$, with

$$\vec{\mathbf{f}} = \begin{bmatrix} < \sum_{i=1}^{4} \omega_i \sum_{j=0}^{n} a_{ij} h_j, h_0 > \\ \vdots \\ < \sum_{i=1}^{4} \omega_i \sum_{j=0}^{n} a_{ij} h_j, h_n > \end{bmatrix} \quad \mathbf{H} = \begin{bmatrix} \sum_{j=0}^{n} < h_j, h_0 > \\ \vdots \\ \sum_{j=0}^{n} < h_j, h_n > \end{bmatrix}$$

$\mathbf{H}$ is an n x n invertible matrix, and thus coefficient vector:

$$\vec{\mathbf{b}} = \mathbf{H}^{-1}\vec{\mathbf{f}} \qquad (4)$$

Notice $\vec{\mathbf{f}}$ can be pre-computed as it involves the integral of predefined basis functions. Further, if basis functions $\{h_j\}$ are chosen to be orthogonal to each other, then the coefficients $\{b_j\}$ can be computed efficiently by simply multiplying a previously calculated matrix with known right hand side. It is therefore possible to model an arbitrary large gridded data set with one matrix multiplication!

6.5 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 $\eta_1 \ge \eta_2 \ge \ldots \ge \eta_l$, 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 $T = (\varepsilon - \mu)/(\sigma/\sqrt{n})$ 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 [19]. If the confidence is 90% and the value T is within the interval [-A, A], we say with 90% confidence:

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

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

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

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

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

6.6 Further Discussion

Boundary continuity: In the DQT hierarchy, a parent node is divided into four quadrants that are completely decoupled from each other, and no continuity conditions are enforced across the quadrants. 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. In DQT, we can adopt a weighted function method to smooth the boundary estimation by introducing certain size of overlapping areas near boundaries. The weight

function, w_I can be selected to guarantee that the global approximation F(.) osculates with $F_I(x,y)$in value and first derivative at centroid of Ω^I. If weighting function is assumed to be polynomial in an independent variable x, then adopting the procedure listed in Ref. [27] the lowest order weight function can be shown to be simply:

$$w(x) = \begin{cases} (1-x^2(3+2x)\,(-1 \le x < 0) \\ 1-x^2(3-2x) \quad (0 \le x \le 1) \end{cases}$$

It has been proved in [27] that using such specially designed weighted functions, the global model can achieve arbitrary order of piecewise continuity. This method can easily be generalized to N dimensional space.

Handling temporal data: We have presented our framework to model spatial data. However, a temporal model can also be incorporated under 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 [29], a theoretical result is given between approximation error and updating frequency.

Handling sequential data: The analysis in our paper assumes data integrity and completeness in the regression process, while in practice data may be only partially available at a moment. For example, if some sensor nodes are mobile, they provide measurements sequentially. It is desirable that clusterheads update their models upon receipt of new corresponding data subset. Linear sequential regression technique [5] can be used to handle this problem.

Supppose $\mathbf{Y}_k$ is the original observed data that follows $\mathbf{Y}_k = \mathbf{X}_k\boldsymbol{\alpha}_k + \boldsymbol{\varepsilon}_k$ in equation (1), and $\mathbf{Y}_{k+1}$ is new data observed. Then we can use the following Kalman Update Equation for computing the new estimation of $\boldsymbol{\alpha}_{k+1}$:

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

$$\mathbf{K}_{k+1} = \mathbf{P}_{k+1}\mathbf{X}_{k+1}^{T}$$

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

$\mathbf{K}_{k+1}$ is kalman gain matrix and $\mathbf{P}^{-1} = (\mathbf{X}^T\mathbf{X})^{-1}$ is so-called information 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.

Multimedia queries: 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 [22] 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 is filtered and extracted of semantically useful information and stored in multi-dimensional vectors. Many feature extraction techniques, such as color histogram and gray scale [13], vector approximation [12], shape indexing [4] 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 (other techniques such as wavelet compression are also applicable in the hierarchical DQT framework) 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.

7 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 two aspects: 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 3200x3200. The distance between each node is 200m, while 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.

7.1 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 node pair is randomly chosen. We use two measurements s and s′, where s is the ratio of the DQT querying cost to the distance 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′ ratio of routing a message from the querying point to the event. We found the

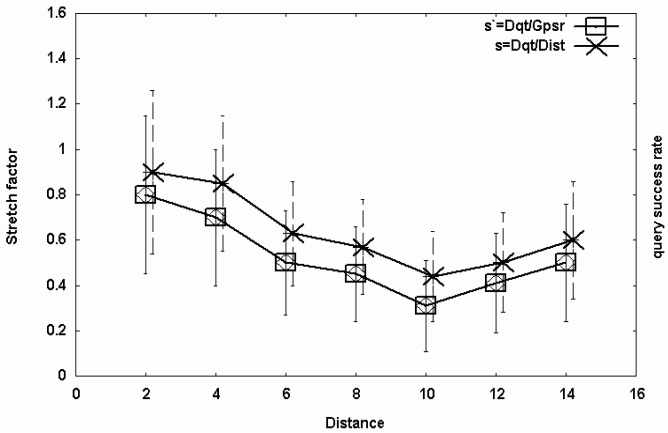

Figure 8: Stretch factor s and s´ with varied query distance

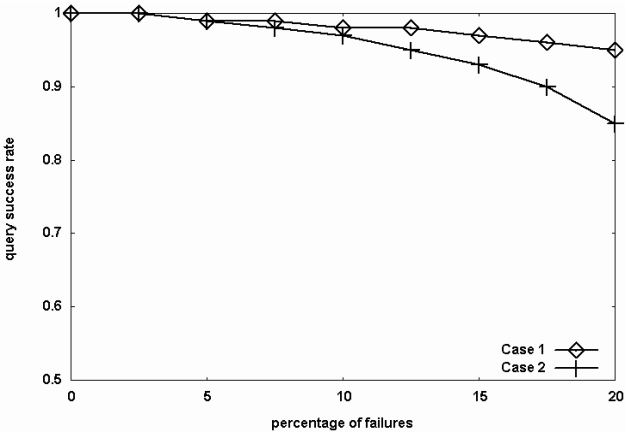

Figure 9: Query success rate for case 1 and case 2

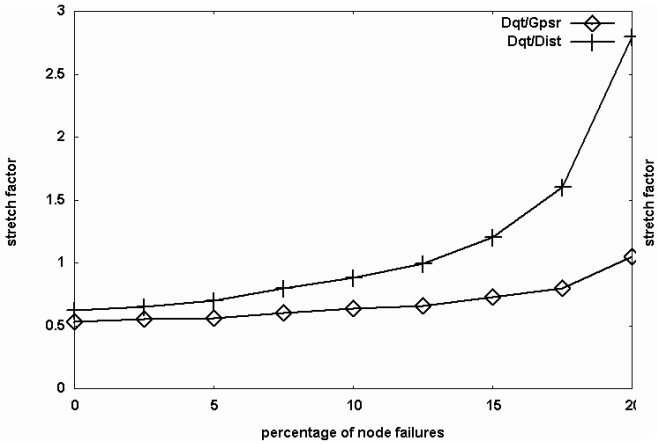

Figure 10: Stretch factor with node failure: Case 1

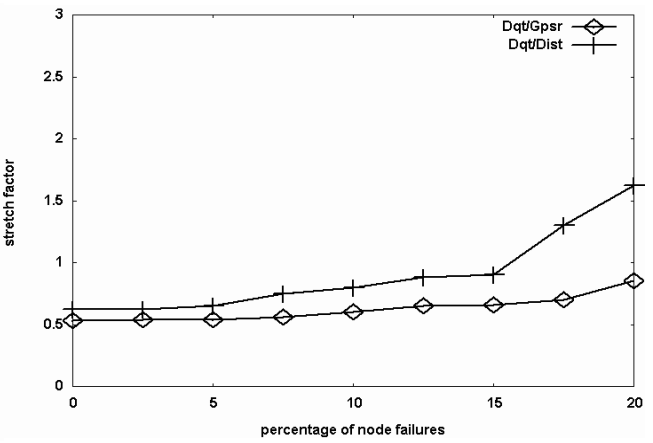

Figure 11: Stretch factor with node failure: Case 2

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 the total. Our scheme has a considerably smaller stretch factor compared to the DSIB scheme, which has an average value 0.9~1.

Fig.8 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´ tend to be 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 to 14/16 etc), 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.

7.2 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.

We keep the same topology and setting in our experiment as section 7.1. We first experiment on the DQT query success rate with node failure. We randomly remove a certain percentage of nodes, 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.9 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.

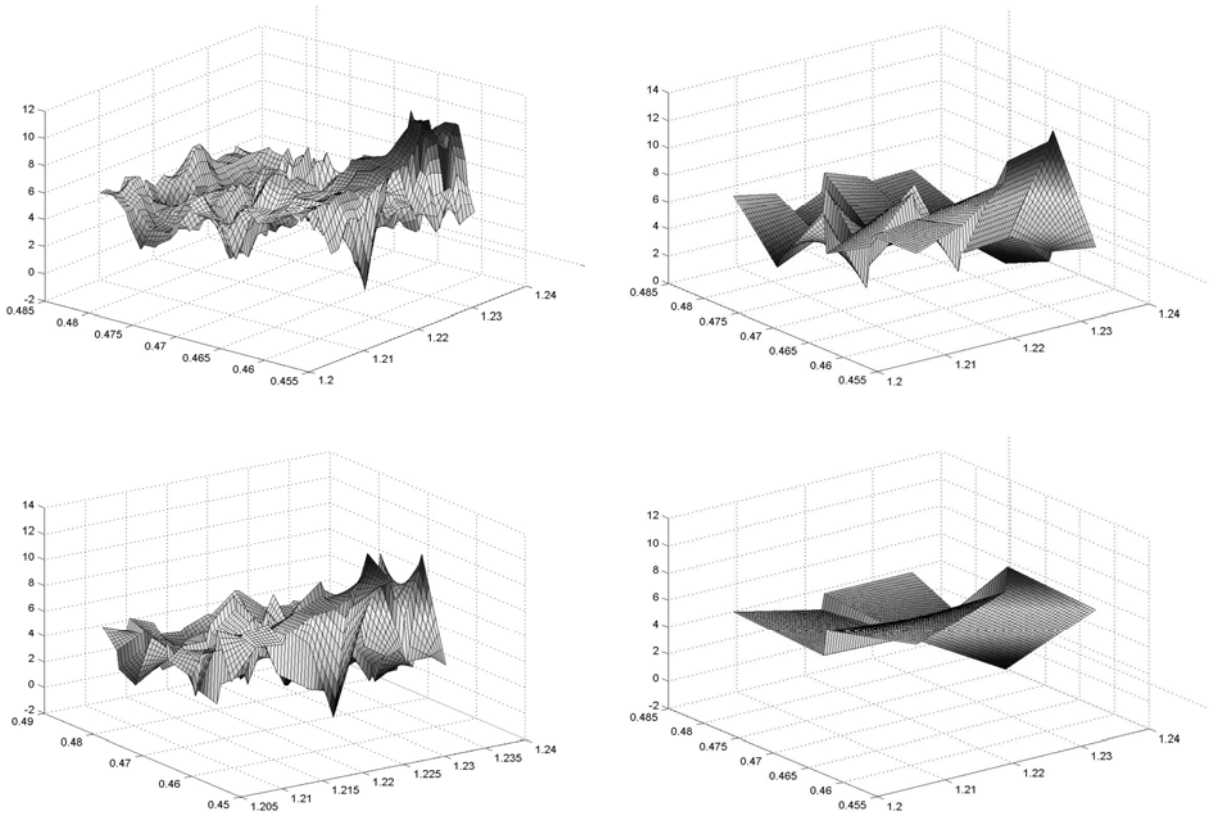

Figure 12: Hierarchical multi-resolution plots: from upleft to downright plots are level 2, 3 4 5 respectively

From the stretch factor point of view, case 2 is also worse than case 1. Fig.10 and Fig.11 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 failure), which increases the cost of searching rapidly. Case 1 performs better than case 2 because in the occurrence of holes, 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.

7.3 Sample Data Analysis for range querying

To verify the range querying algorithms in modeled DQT, we use sample dataset from PRISM group [30]. 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 temporatures. The resolution of this sample data is 0.0417decimal degrees both in Latitude and Longi- tude, which is about 4km. The map total covers a region 256x256 km^2. 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 derivedof 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. 12 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. 13 with 80%, 90% and 95% confidence 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 more

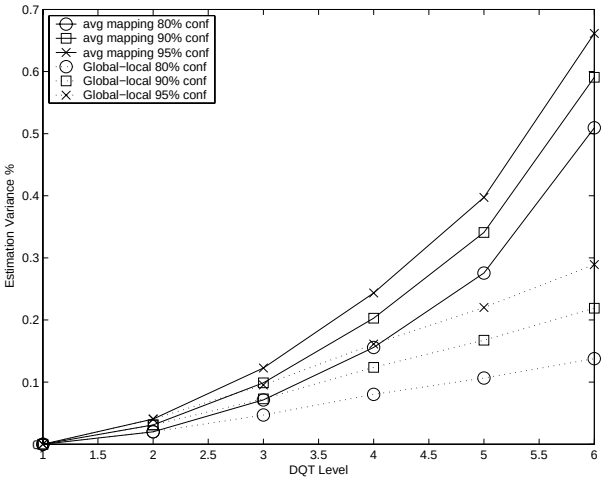

Figure 13: Mean approximation variance (global-local algorithm)

sharply than global-local algorithm. The reason is that average-mapping only considers the mapped nodes and lost 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% confidence 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% confidence, level 3 would be enough for answering queries using global-local algorithm. For each specific level, higher confidence leads to larger approximation variance. There are two possible ways to handle an unsatisfied query: either by reducing the confidence 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 ex amples for two types of range querying: lookup querying and searching. All queries assume 10% approximation-variance with 90% confidence. Fig. 14a considers 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. 14b 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 6) with naive querying scheme (as in Fig 5) by assuming the query originator's ID 00000.

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

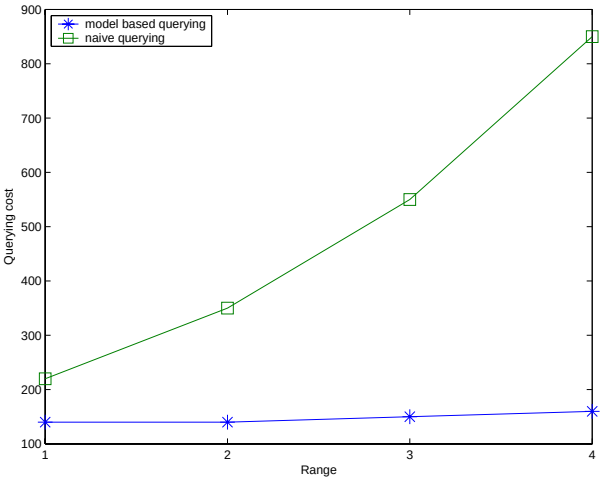

Figure 14a: Lookup querying cost comparison

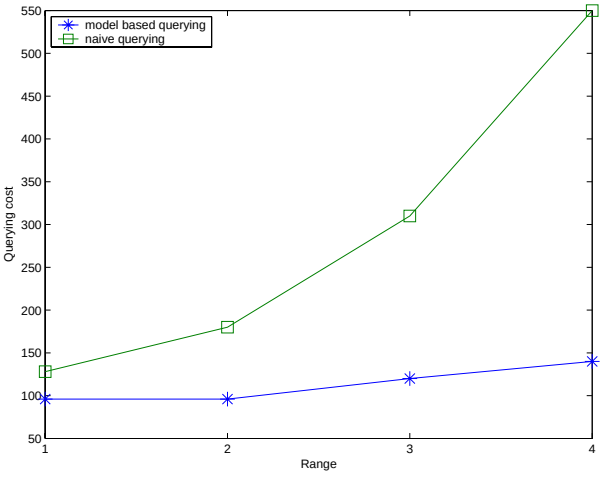

Figure 14b: Searching querying cost comparison

the range. For searching problems, model based approach can filter nodes at high levels in favor of models, thus performs more efficiently than naïve 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.

8 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, R. Ramnath, E. Ertin, and P. S. et. al., Exscal: Elements of an extreme scale wireless sensor network, in 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), 2005.

[2] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, et al., A Line in the Sand: A Wireless Sensor Network for Target Detection, Classification, and Tracking, Computer Networks, Vol. 46, Issue 5, pp. 605-634, Dec. 5, 2004.

[3] P. B. Gibbons. Distinct sampling for highly-accurate answers to distinct values queries and event reports. In Proc. of VLDB, Sept 2001.

[4] J. S. Beis, D.G. Lowe, Shape Indexing Using Approximate Nearest-Neighbour Search in High-Dimensional Spaces, CVPR, p. 1000, 1997.

[5] J. L. Crassidis and J. L. Junkins, "Optimal Estimation of Dynamic Systems", Chapman & Hall/CRC (April 23, 2004).

[6] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong. Model-Driven Data Acquisition in Sensor Networks. In VLDB Conference, September 2004

[7] M. Demirbas, A. Arora, and M. Gouda, Pursuer - Evader Tracking in Sensor Networks. Sensor Network Operations, chp.9, IEEE Press, May 2006.

[8] 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), France, December 2004.

[9] M. Demirbas, X.Lu, Distributed Quad-Tree for Spatial Querying in Wireless Sensor Networks, In Proceedings of ICC, 2007.

[10] R. Finkel and J.L. Bentley , Quad Trees: A Data Structure for Retrieval on Composite Keys. Acta Informatica ,1974, 4 (1): 1-9.

[11] S. Funke, L. J. Guibas, A. Nguyen, and Y. Wang. Distance sensitive routing and information brokerage in sensor networks. In DOCSS, 2006.

[12] A. Gersho, Vector Quantization and Signal Compression, Kluwer Academic Publishers, 1992.

[13] C. Guestrin, P. Bodik, R. Thibaux, M. Paskin, S. Madden, Distributed Regression: an Efficient Framework for Modeling Sensor Network Data. In Proceedings of IPSN, 2004.

[14] D. Ganesan, D. Estrin, J. Heidemann. DIMENSIONS: Why do we need a new Data Handling architecture for Sensor Networks? In Proceedings of the ACM Workshop on Hot Topics in Networks, pp. 143-148. October, 2002

[15] L. G. Shapiro and G. C. Stockman "Computer Vision" Prentice Hall, 2003.

[16] C. Intanagonwiwat, R.Govindan ,D.Estrin, Directed diffusion: a scalable and robust communication paradigm for sensor networks, Proceedings of the 6th annual international conference on Mobile computing and networking, p.56-67, August 06-11, 2000

[17] 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), August 1999.

[18] B. Karp and H. T. Kung. Gpsr: greedy perimeter stateless routing for wireless networks. In MobiCom: p243–254, 2000.

[19] S. Kotz and S. Nadarajah, Multivariate T distributions and their applications, Cambridge University Press, 2004

[20] 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 Homes), Vol. 9, No. 6, pp. 77-84, Dec 2002

[21] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. ACM SIGMOD, Int. Conf. on Management of Data, San Diego, June 2003

[22] M. Rahimi, R. Baer, O. Iroezi, J. Garcia, J. Warrior, D. Estrin, M. Srivastava, Cyclops: in situ image sensing and interpretation in wireless sensor networks, In SenSys 2005.

[23] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. Ght: a geographic hash table for datacentric storage. In WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, pages 78–87, 2002.

[24] L.Savidge, H. Lee, H.Aghajan, A.Goldsmith, QoS-based geographic routing for event-driven image sensor networks, In, Proc. of BaseNets, October 2005.

[25] R. Szewczyk, A. Mainwaring, J. Polastre and D. Culler, An Analysis of a Large Scale Habitat Monitoring Application, Sensys, 2004.

[26] P. Singla, "Multi-Resoultion Methods For High Fidelity Modeling and Control Allocation in Large Scale Dynamical Systems," Texas A&M University, May 2005.

[27] P. Singla and J. L. Junkins, "Global Local Orthogonal MAPping (GLO-MAP) in N-Dimensions: Applications to I/O Approximation," Invited Paper, 6th Conference on Dynamics and Control of Systems and Structures in Space, Cinque Terre, Italy, July-2004.

[28] G. Tolle, J. Polastre, et.al, A Macroscope in the Redwoods, In Proceedings of SenSys, November 24, 2005.

[29] 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, 2004

[30] M. Widmann and C. Bretherton. 50 km resolution daily precipitation for the Pacific Northwest, http://www.ocs.oregonstate.edu/prism/

[31] C. Yu and W. Meng. Principles of Database Query Processing for Advanced Applications. Morgan Kaufmann, San, 1998.
[32] Y. Yao and J. Gehrke. Query processing in sensor networks. In Conference on Innovative Data Systems Research (CIDR), 2003.