**Title**

# Energy-Efficient Data Organization and Query Processing in Sensor Networks



**Authors**
Gummadi, Ramakrishna
Li, Xin
Govindan, Ramesh
et al.

**Publication Date**
2005-05-05

Peer reviewed

# Poster Abstract: Energy-Efficient Data Organization and Query Processing in Sensor Networks

Ramakrishna Gummadi gummadi@usc.edu
Xin Li xinli@usc.edu
Ramesh Govindan ramesh@usc.edu
Cyrus Shahabi cshahabi@usc.edu
Wei Hong wei.hong@intel.com

## ABSTRACT

Recent sensor networks research has produced a class of data storage and query processing techniques called *Data-Centric Storage* that leverages locality-preserving distributed indexes like DIM, DIFS, and GHT to efficiently answer multi-dimensional range and range-aggregate queries. These distributed indexes offer a rich design space of a) logical decompositions of sensor relation schema into indexes, as well as b) physical mappings of these indexes onto sensors. In this poster, we explore this space for energy-efficient *data organizations* (logical and physical mappings of tuples and attributes to sensor nodes) and devise *purely local query optimization techniques* for processing queries that span such decomposed relations. We propose four design techniques: (a) fully decomposing the base sensor relation into distinct sub-relations, (b) spatially partitioning these sub-relations across the sensornet, (c) localized query planning and optimization to find fully decentralized optimal join orders, and (d) locally caching join results. Together, these optimizations reduce the overall network energy consumption by *4 times* or more when compared against the standard single multi-dimensional distributed index on a variety of synthetic query workloads simulated over both synthetic and real-world datasets. We validate the feasibility of our approach by implementing a functional prototype of our data organizer and query processor on Mica2 motes and observing comparable message cost savings.

**Categories and Subject Descriptors:** H.4 [Information Systems Applications]: Miscellaneous

**General Terms:** Design, Performance, Experimentation

**Keywords:** Sensor Networks, Data-Centric Storage, DCS, Data Organization, Query Optimization, Caching

## 1. INTRODUCTION AND MOTIVATION

Wireless sensor networks are an emerging class of highly distributed systems with widespread applicability. In such networks, nodes generate, process and store sensor readings within the network. This architecture is necessitated by the relatively high energy cost of wireless communication—this cost makes it infeasible to consider centrally collecting and processing voluminous sensor data. An important component of these networks, then, is an energy-efficient system that enables users to query the stored data.

Existing approaches to organizing data and processing queries fall under one of the two broad categories namely, *Data-Centric Routing* (DCR) and *Data-Centric Storage* (DCS). In DCR, the data generated by the sensors is stored at the nodes that generate them, and queries are flooded throughout the network. Data from the sensors in the sensornet is then aggregated along the query tree that is built during the query flooding phase on a per-query basis. This approach, pioneered by early systems such as TinyDB [5] and Cougar [1], is efficient for *continuous* (long-running) queries, where the high energy cost incurred during the query flooding and per-query data aggregation phases is amortized over time.

DCS is a relatively new class of data storage and query methodologies proposed in [7]. In DCS, data generated by a sensor is first stored *intelligently* at *remote* nodes as soon as it is generated. This is done with an eye toward exploiting data locality during querying because *related* sensor data gets stored together *regardless* of where in the sensornet the data originates. Consequently, queries can be directed to *precise data locations* of the network during the query propagation phase *without flooding*, and, data can be *efficiently* and *locally* aggregated during the query processing phase. Thus, the *overall* (insertion+query) cost for DCS is lower than for DCR for many *ad-hoc* (short-lived) workloads.

DCS can use any locality-preserving geographically distributed index structure such as DIM [4], GHT[6], DIFS [3], and DIMENSIONS [2]. In this poster, our focus is to examine techniques that improve the overall energy performance of vanilla DCS based on such a family of distributed indexes:

1. We exploit the *flexibility* offered by these data structures to derive better performing *data organizations* (mappings of tuples and attributes to network nodes) compared to the naïve and rigid mapping used today.
2. We study decentralized and high performance *query planning and optimization* in such DCS systems.

## 2. APPROACH

Consider a sensor network with an *m*-relation schema $\langle uuid, a_1, a_2, \ldots, a_m \rangle$. Such a relation schema is called a base relation. Tuples in this schema can be stored in one DIM. Alternatively, we can fully decompose them into *m* DIM's each of which stores a single relation of the form $\langle uuid, a_i \rangle$, and we can then *join* on *uuid* on demand to evaluate queries. A spectrum of partial decompositions of the base relation into sub-relations of the form $\langle uuid, a_i, \ldots, a_j \rangle$ is, of course, also conceivable. Clearly, we can expect these different data organizations to yield different performance under different workloads. Our measure of performance is the total energy cost incurred for a given workload, including data inserts and query retrievals. (We approximate the energy cost of a single message as a product of the size of the message (in bits) and the number of hops the message traverses.)

In this work, we want to *analytically* and *experimentally* explore the design space of data storage and query processing in sensornets. We use DIM's [4] as the distributed index for our base storage, in-


*SenSys'04,* November 3–5, 2004, Baltimore, Maryland, USA.
ACM 1-58113-879-2/04/0011.

dexing, and querying layer. We found that, in many cases, *fully decomposing* the base relation performs better than zero or any partial decomposition, even if the decomposition is carried out with an eye toward a given query workload. We also propose three related mechanisms that can improve the efficiency of query processing when a base relation is fully decomposed into multiple DIM's:

**Spatially Partitioning Sub-Relations** Each fully decomposed sub-relation is stored in a DIM, and all DIM's are assigned spatially disjoint sections of the sensor field.

**Efficient Query Planning via Decentralized Join-Ordering** We import the database notion of *joins* into sensornets for range and range-aggregate query processing within our system. Our system compiles an SQL query at the query issuer and constructs an *efficient query plan* that includes an *optimal* join order using only locally available information in the form of a histogram. We argue that it is important to choose a *good join order* during query optimization and demonstrate that it is possible to do so using only summarized global information in the form of a low overhead coarse-grained multi-dimensional histogram that approximates the distribution of data stored within the network. Our query optimizer computes the total query energy cost as:

$$E = |\bar{a_1}| \times D(\bar{a_1},\bar{a_2}) + |J(\bar{a_1},\bar{a_2})| \times D(\bar{a_2},\bar{a_3}) + |J(\bar{a_1},\bar{a_2},\bar{a_3})| \times D(\bar{a_3},\bar{a_4})| + \ldots + |J(\bar{a_1},\ldots,\bar{a_{k-1}})| \times D(\bar{a_{k-1}},\bar{a_k}) + |J(\bar{a_1},\ldots,\bar{a_k})| \times \Sigma_{\bar{a_k}} \quad (1)$$

where $\bar{a_i}$ denotes the query range on attribute $a_i$, $|\bar{a_1}|$ denotes the number of tuples that would be produced by the first step of the range selection, $D(\bar{a_i},\bar{a_j})$ denotes the average distance between the nodes in the DIM's containing $\bar{a_i}$ and $\bar{a_j}$, and $\bar{a_j}$), and $|J(\bar{a_1},\bar{a_2},\ldots,\bar{a_i})|$ denotes the number of tuples that would be produced after the *i*th join operation in the query. In particular, $|J(\bar{a_1},\bar{a_2},\ldots,\bar{a_{k-1}})|$ denotes the number of tuples that would be produced by $k-1$ joins before the final join step; this latter yields $|J(\bar{a_1},\ldots,\bar{a_k})|$ tuples that need to be aggregated within DIM $DIM_{a_k}$ for an average aggregation cost per tuple of $\Sigma_{\bar{a_k}}$. The query optimizer then picks a join order that minimizes this energy cost. The various terms in the above equation are *estimated* using the histogram and query selectivity factors.

**Efficient Query Execution via Optimistic Join-Caching** We propose a *simple* and *robust* mechanism to locally cache the results of partial joins across sub-relations at each sensor node. This caching strategy enhances query performance by eliminating redundant tuple movement during query execution.

# 3. EVALUATION

## 3.1 Simulation

We evaluated the performance of our approach using simulations over both real-world (Great Duck Island) and synthetic datasets on a wide variety of query workloads. We observed a performance benefit well over a *factor of 4* compared to the base case of a single full-dimensional DIM. We also compared our performance against schemes that do not perform join order optimization, and those that do not perform join tuple caching to understand the *individual* performance contribution of each of the techniques mentioned in Section 2.

In Figure 1, we compare the bit energy performance of our full scheme (called optimized), our scheme with random join ordering (called random), our scheme with worst-case join ordering (called worst), base case single vanilla DIM on four attributes (called 4-DIM), and our scheme without join caching (called uncached).

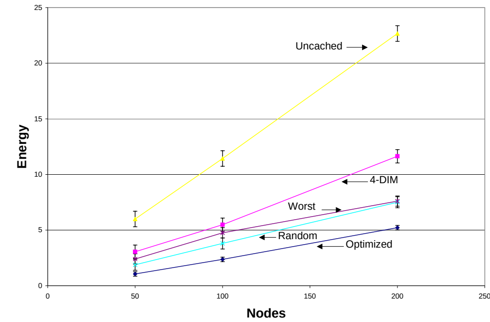


**Figure 1: Bit-energy costs of 4-DIM, Optimized, Random, Worst, and Uncached for with 100 queries**

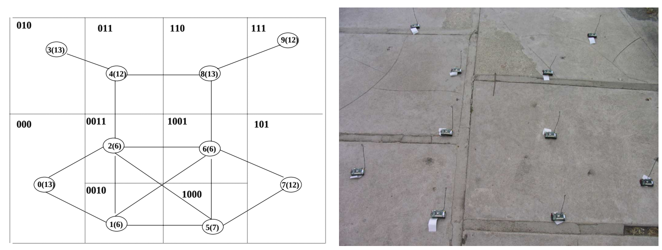


**Figure 2: Topology used for Implementation**

## 3.2 Implementation

In Figure 2, we show the topology we used for an implementation prototype on Mica2 motes in which we observed a performance benefit of $2.7X$ with just two attributes.

# 4. REFERENCES


[1] P. Bonnet, J. E. Gerhke, and P. Seshadri. Towards Sensor Database Systems. In *Proceedings of the Second International Conference on Mobile Data Management*, January 2001.

[2] D. Ganesan, D. Estrin, and J. Heidemann. DIMENSIONS: Why do we need a new Data Handling architecture for Sensor Networks? In *Proc. HotNets-I*, Princeton, NJ, October 2002.

[3] B. Greenstein, D. Estrin, R. Govindan, S. Ratnasamy, and S. Shenker. DIFS: A Distributed Index for Features in Sensor Networks. In *Proc. IEEE WSNPA*, Anchorage, AK, May 2003.

[4] X. Li, Y. J. Kim, R. Govindan, and W. Hong. Multi-dimensional Range Queries in Sensor Networks. In *Proc. Sensys*, Los Angeles, CA, November 2003.

[5] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. In *Proc. ACM SIGMOD*, pages 491–502. ACM Press, 2003.

[6] 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 *Proc. WSNA*, Atlanta, GA, September 2002.

[7] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin. Data-centric storage in sensornets. *SIGCOMM Comput. Commun. Rev.*, 33(1):137–142, 2003.