

Reliable Hierarchical Data Storage in Sensor Networks

Song Lin – (slin@cs.ucr.edu)
Benjamin Arai – (barai@cs.ucr.edu)
Dimitrios Gunopulos – (dg@cs.ucr.edu)
Gautam Das – (gdas@cse.uta.edu)

<http://www.cs.ucr.edu/~barai>

Background

- Retrieving sensor data can be classified into the following categories:

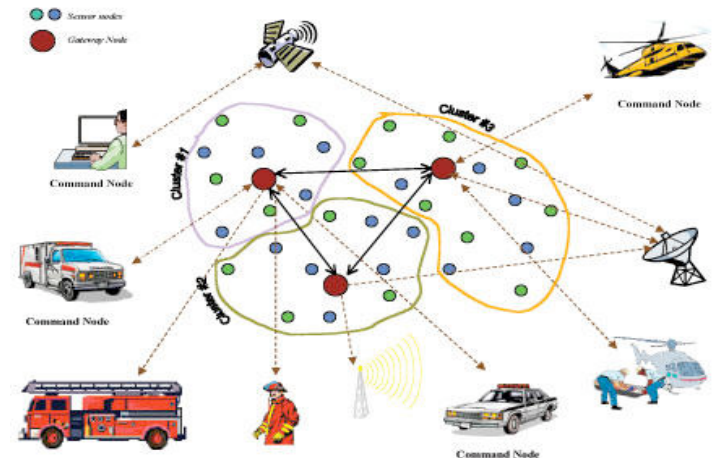
- **Local storage**

- Keep data stored a nodes locally
- To retrieve data a query floods the network

- **External storage**

- Data sent to base station without waiting for query
- Waste energy transmitting data

- **Data-centric storage**



Motivation

- Data-centric storage scheme reduces the number of messages sent over the network by storing messages with little impact at individual sensors for later analysis.
- Recent data-centric storage algorithms in sensor networks usually require geographical location, which is not always available.
- The reliability of the data stored within the sensors is another concern in failure prone sensor networks.

Our Contributions

- A hierarchical storage system where all sensors cooperate in storing data into a single database.
- A resilient data transmission mechanism to work against link failures during the routing.
- A data reliability algorithm for recovering data from sensor failures with a self-mending mechanism.
- An efficient load balancing algorithm to equalize the amount of data stored in individual sensors in such a way that energy expended at individual sensors is balanced.

Problem Formulation

- Given a sensor network of n sensors where each sensor S_i has a neighbor list indicating the set of sensors that S_i can communicate directly. The goal is to segment the domain D of the sensor data into several non-overlapping intervals and assign different intervals to different sensors, such that we can efficiently answer the Equality Queries and Range Queries from any sensor in the network.
- **Equality Query:** a one dimensional query $Q(v, a)$ in which the attribute v in the record r is equivalent to value a .
- **Range Query:** a one dimensional query $Q(v, lb, ub)$ in which the attribute v in the record r is between the lower bound lb and upper bound ub . The Equality Query is a special case of the Range Query $Q(t, lb, ub)$ in which $lb = ub$.

Related Work

- Data Centric Storage
 - GHT¹: geographical Information is required
 - GEM²: vulnerable to failures,
 - Only indexes and stores data at leaf nodes
 - Does not provide reliable data storage against failures
- Reliable Storage
 - R-DCS³: requires a global view of the network topology along with the position of each sensor
- Load Balancing
 - Backbone Tree⁴, Node-Centric⁵: centralized, requires large network traffic to update

•Data-centric storage in sensornets with GHT, a geographic hash table. *Mobile Network Applications*, 2003.

• Gem: graph embedding for routing and data-centric storage in sensor networks without geographic information. *Sensys*, 2003.

• Resilient data-centric storage in wireless ad-hoc sensor networks. *Mobile Data Management*, 2003.

• Load-balancing routing for wireless access networks. *INFOCOM*, 2001.

• A node-centric load balancing algorithm for wireless sensor networks. *GLOBECOM*, 2003.

Indexing Sensor Data

- Interval Tree, a reliable and load balancing data-centric storage algorithm for sensor networks
 - Interval Distribution
 - Resilient Routing
 - Reliable Storage
 - Load Balancing

Tree Construction

- Randomly assign *Root* and send tree construction message
- Each node computes (self-count) and (child-count)

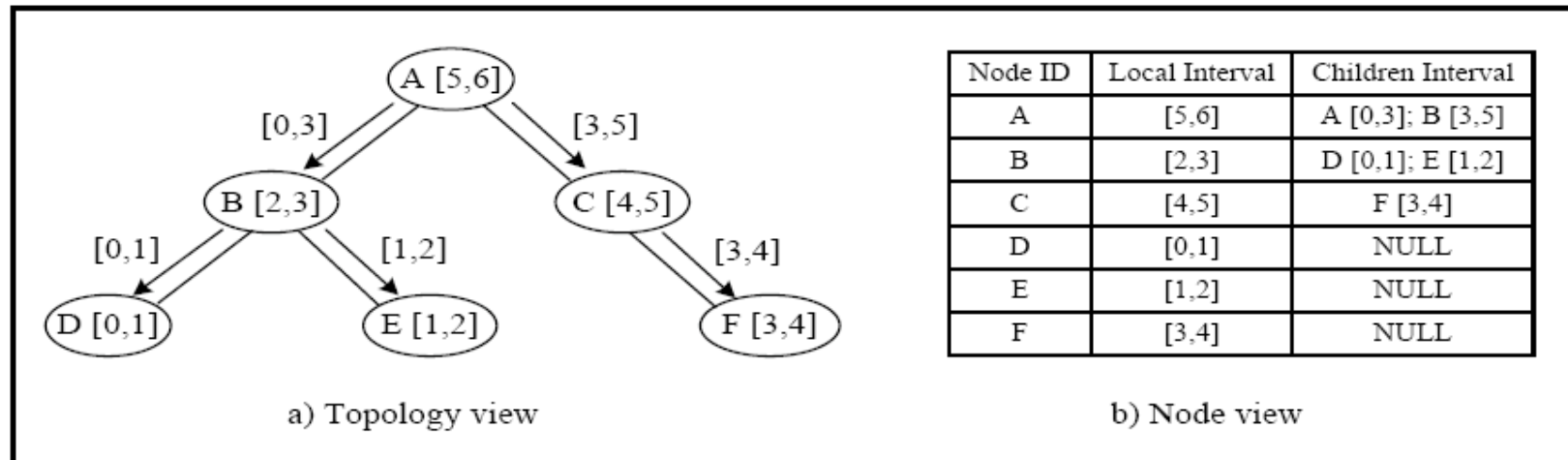


Figure 1. The *Interval Tree*

Interval Distribution

- Given:
 - The domain of the data records $D = [LB, UB]$ (e.g., $0^\circ\text{F} - 150^\circ\text{F}$ for temperature records)
 - n sensors in the network, we evenly segment D into n non-overlapping intervals.

$$I_j = \left[LB + \frac{UB - LB}{n} \times j, LB + \frac{UB - LB}{n} \times (j + 1) \right], (0 \leq j \leq n - 1)$$

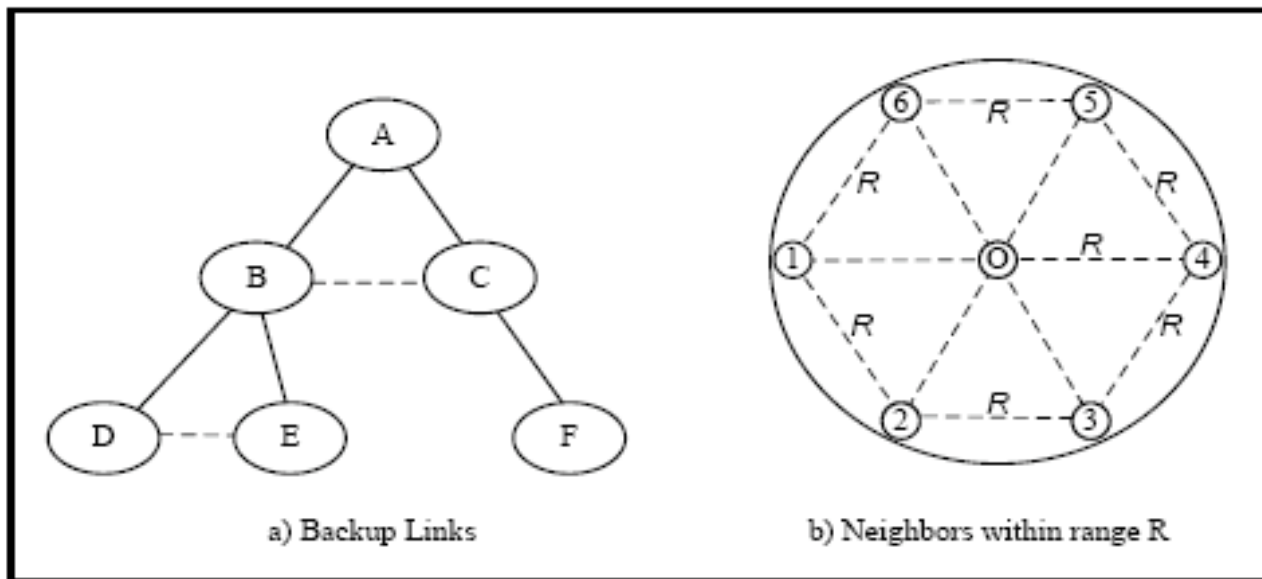
- Starting from the Root, each node locally indexes the last one of the given intervals (Local Interval) and assigns the other intervals to its children proportionally to the child-counts at different children.
- We keep assigning intervals level-by-level from the Root towards the leaf nodes of the tree until all the sensors have been assigned their corresponding intervals.

Routing in the Interval Tree

- After the interval distribution to all the sensors, we can start anywhere in the network and route to the query interval by traversing the tree.
- Starting from the query node, the idea is to forward the query upward the tree until finding an overlapping between the query interval and the children interval of some node, and then forward the query to the overlapped child downward the tree until we reach the sensor that indexes the interval.
- *Lemma: The communication distance to find an interval in the Interval Tree is at most $2L$, where L is the height of the tree.*

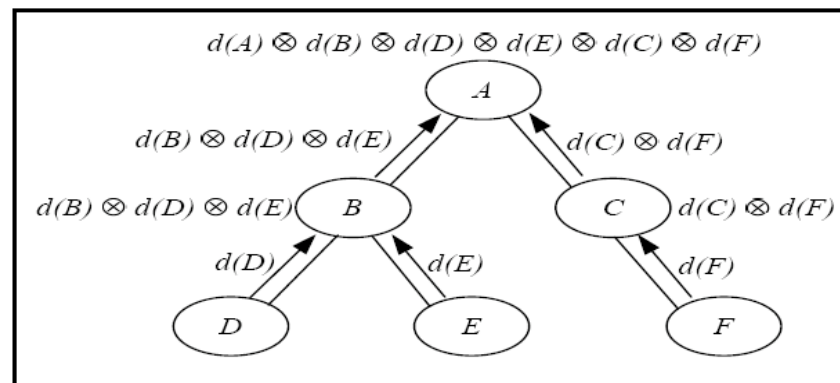
Resilient Data Transmission

- Link Failures: Backup links
 - When some node or some communication link fails in the network, its children detect the failure and utilize the backup links to route the message to the other part of the network



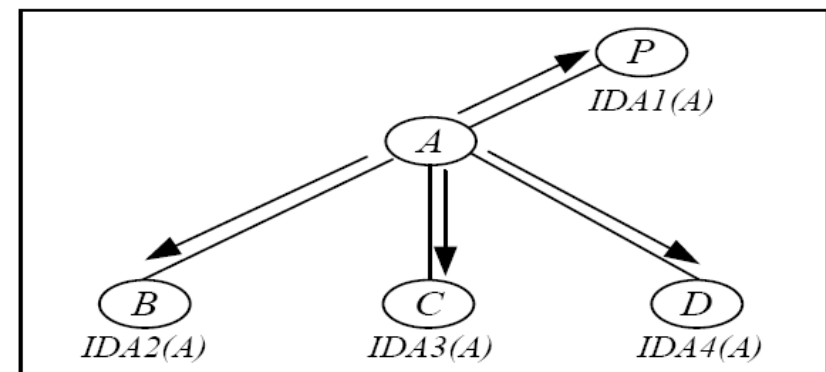
Reliable Data Storage

- XOR BACKUP
 - Inexpensive bit-wise XOR computation
 - Failure Recovery:
 - Suppose a leaf node D fails, it can be recovered using the backup information from B and E : $D = (B \otimes D \otimes E) \otimes B \otimes E$
 - Suppose the internal node B fails, it can be recovered using A , C , D , and E : $B = (A \otimes B \otimes C \otimes D \otimes E \otimes F) \otimes A \otimes (C \otimes F) \otimes D \otimes E$



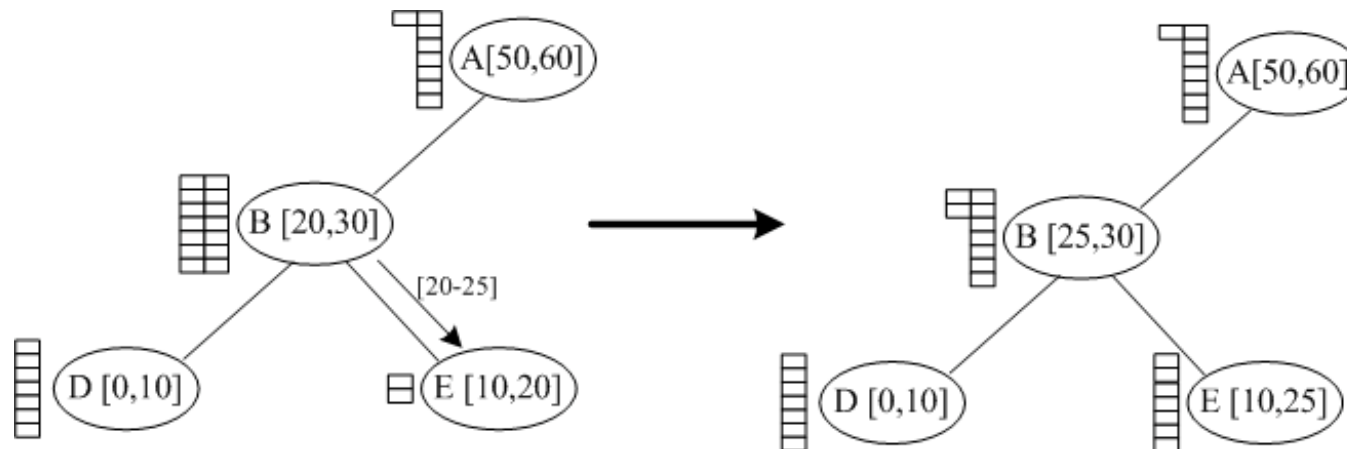
Reliable Data Storage

- IDA BACKUP
 - Information Dispersal Algorithm
 - Distributes data of size L over N even-sized shares (of size L/M each) where any subset of M of these N shares can recover the original data
 - Recover from Multiple Failures
 - $M = N - f$ where f = maximum number of simultaneous failures
 - Cheap Storage Overhead
 - To recover L from $N - M$ failures, we need LN/M redundant information



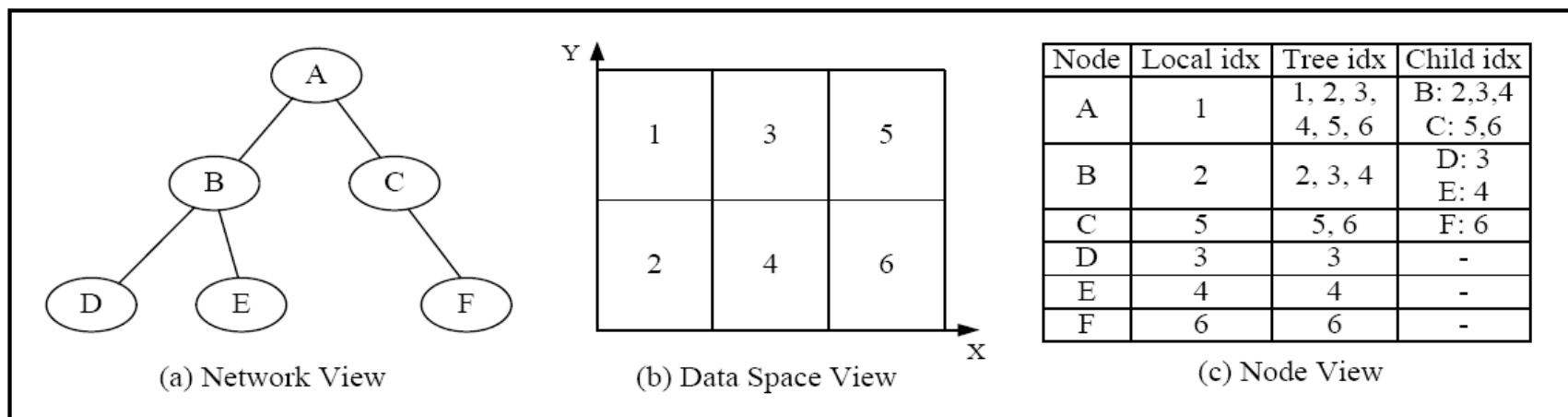
Load Balancing

- Equal sized intervals does not guarantee equal storage load between sensors
- Dynamically adjust the interval length stored at individual nodes to accommodate the biased distribution of the sensor data
 - Allow sensors to communicate with neighbors the size of the data it stores (i.e., children & parent)
 - If difference between two sensors (say S_1 and S_2) then bipartition the popular interval and redistribute the interval



Indexing Multidimensional Data

- Segment the data domain D^t into n non-overlapping and even-sized sub-regions
- Each node S , once receives the sub-regions assigned from its parent, assigns these sub-regions to its direct children proportionally to the cardinalities of the sub-trees rooted from them.



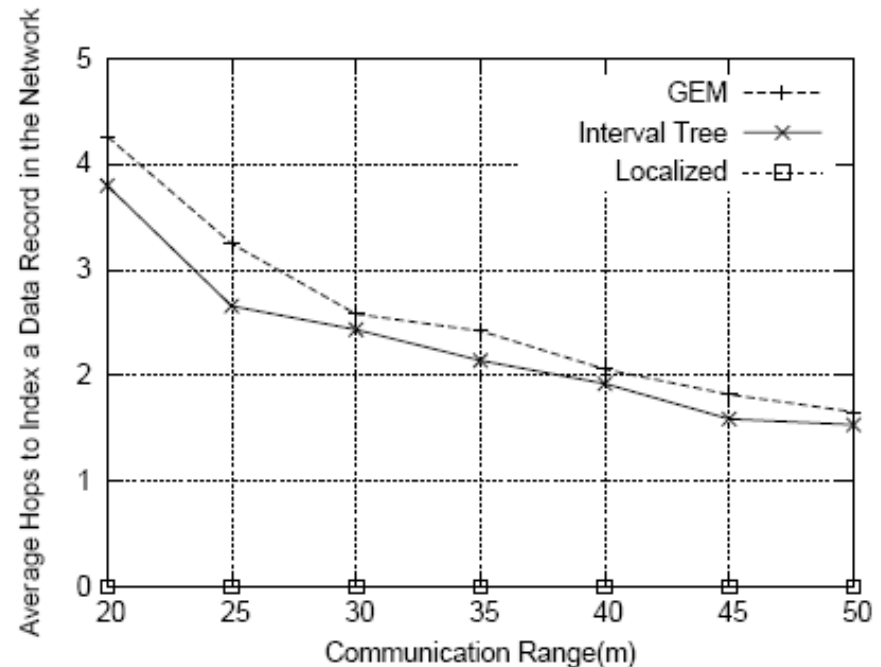
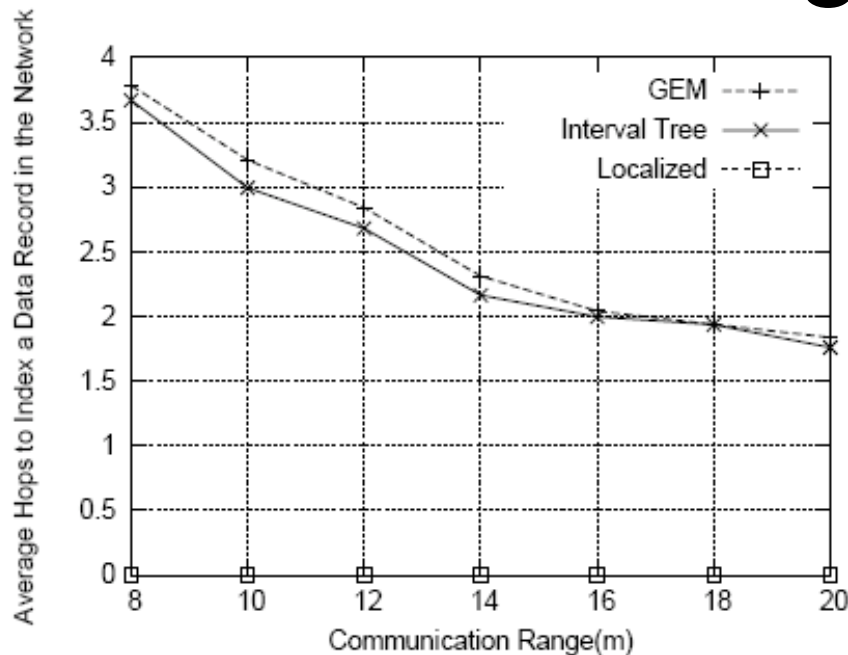
Experimental Evaluation

- Data Sets
 - **Intel:** This real world data set is a trace of sensor readings collected from 54 sensors deployed in the Intel Berkeley Research lab between February and April 2004.
 - **Random:** There are a total of 100 sensors distributed evenly in a 100×100 grid. We adjust the network density and topology by changing the communication range of a sensor. To simulate the real sensor measurements at different locations, we generate both random and biased distribution of the data records at each sensor.

Comparison System

- **GEM:** This is the graph embedded routing and data storage mechanism proposed in [12]. The GEM algorithm constructs a labeled graph that is embedded into the network topology in a distributed fashion.
- **Localized:** This is a simple data storage scheme that stores data where they are generated without indexing them. It introduces no communication overhead for data storage

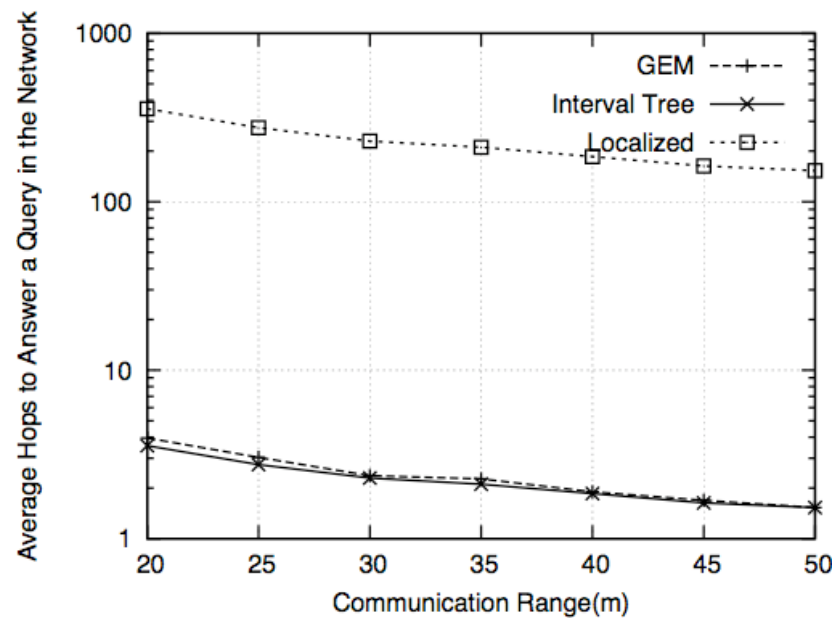
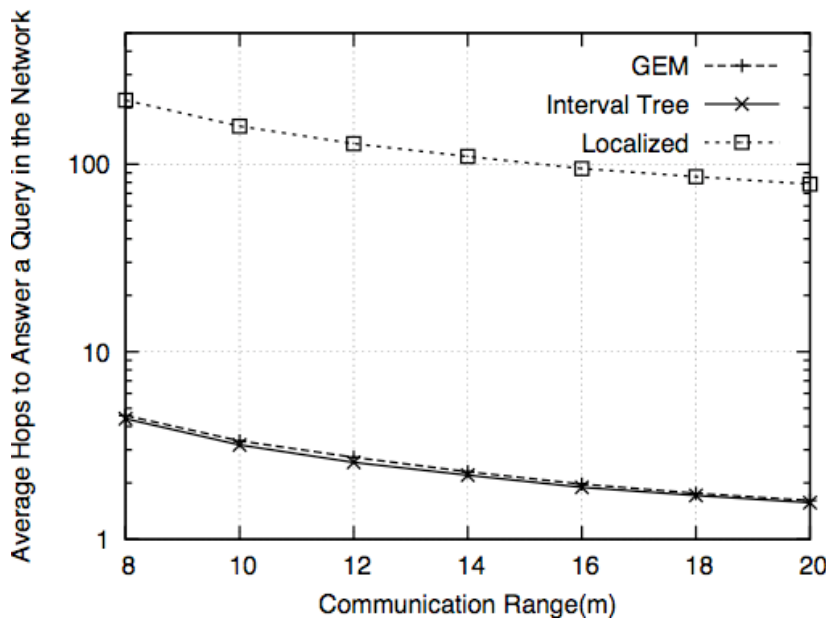
Indexing Overhead



Communication overhead for indexing (Intel data and Random data)

Varying the communication range of individual sensors to change the topology of the sensor network (larger communication range → denser sensor network).

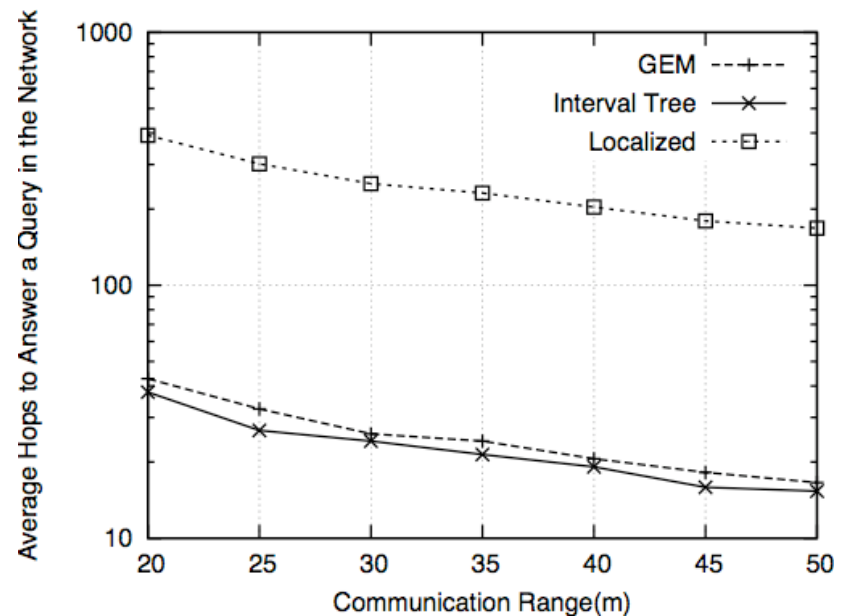
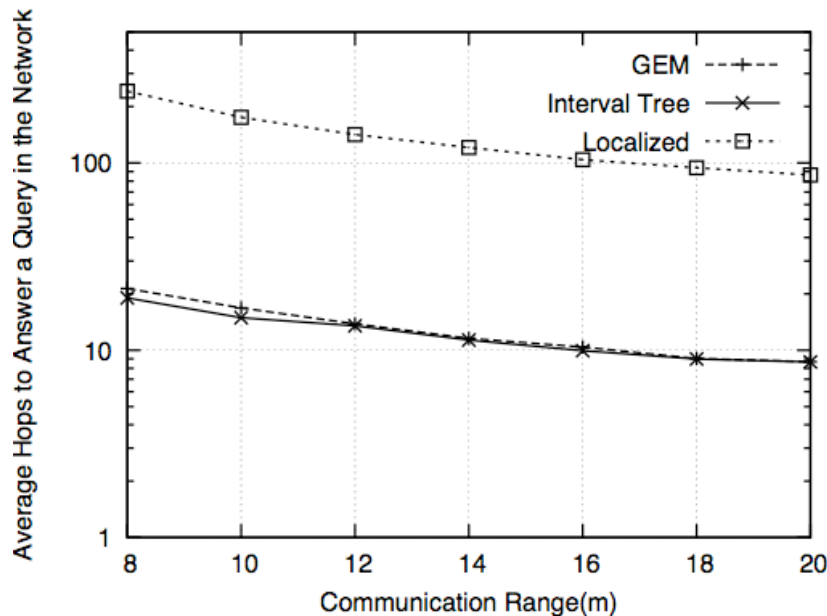
Query Processing Overhead (Equality)



We generate 10,000 random *Equality Queries* and *Range Queries* on the data stored in the sensor network (Intel data and Random data).

Each query is issued from a random sensor in the network and we measure the average communication overhead for each query.

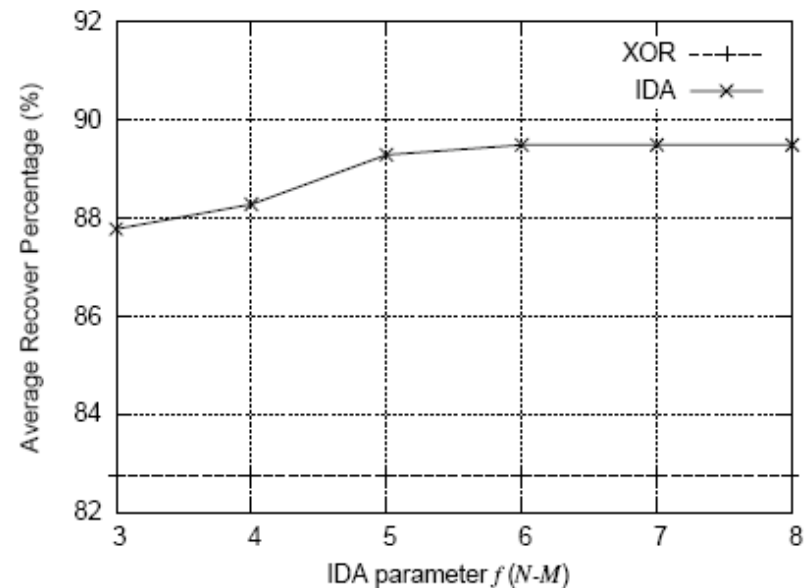
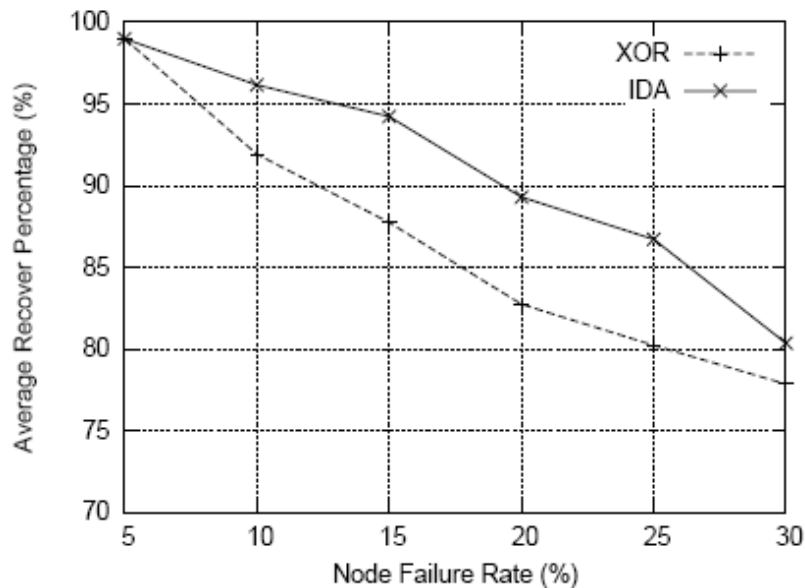
Query Processing Overhead (Range)



We generate 10,000 random *Equality Queries* and *Range Queries* on the data stored in the sensor network (Intel data and Random data).

Each query is issued from a random sensor in the network and we measure the average communication overhead for each query.

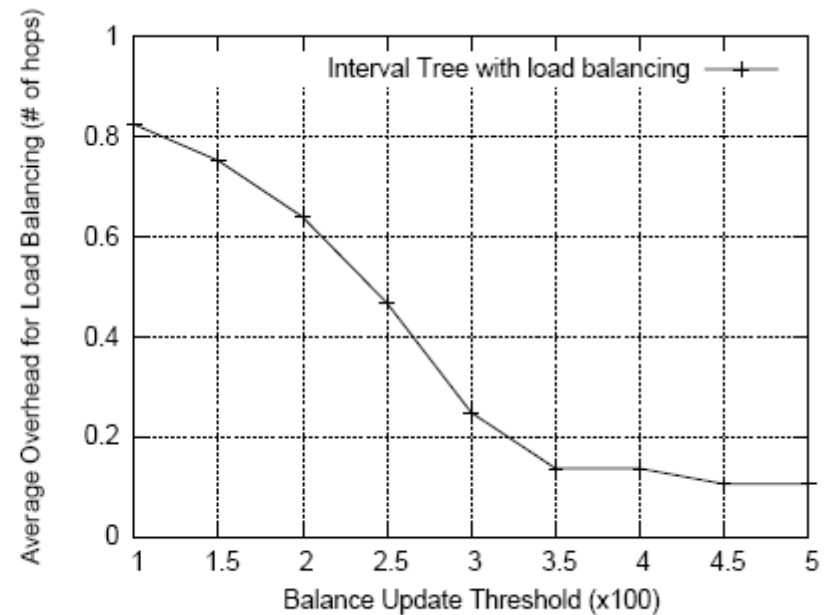
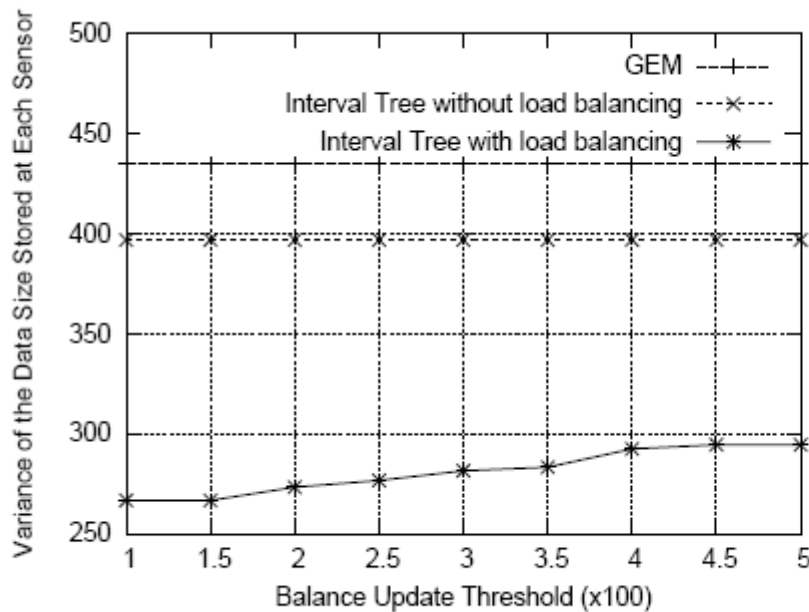
Failure Recovery



We evaluate and compare the performance of two data recovery techniques: the *XOR* and *IDA* algorithms (*Intel Data*)

IDA provides better recovery capacity but requires more maintenance overhead ($f = M - N$)

Load Balancing



Balance the data size stored at different sensors.

With increasing update threshold, the load balancing shows less balancing power but also introduces less overhead (in terms of communication).

Conclusion & Future Work

- Introduced a reliable and load balancing data-centric storage solution for sensor networks
- We proposed a data-centric storage and query routing algorithm, Interval Tree.
- Resilient data transmission mechanisms, and reliable storage of the sensor data.
- Adaptive load balancing mechanism that is able to adjust the interval length indexed at different sensors according to the distribution of the new generated sensor data.

Reliable Hierarchical Data Storage in Sensor Networks

Song Lin – (slin@cs.ucr.edu)
Benjamin Arai – (barai@cs.ucr.edu)
Dimitrios Gunopulos – (dg@cs.ucr.edu)
Gautam Das – (gdas@cse.uta.edu)

<http://www.cs.ucr.edu/~barai>