

In-Network Processing

Presented by:
Mirko Montanari
Riccardo Crepaldi

[Data in Sensor Network]

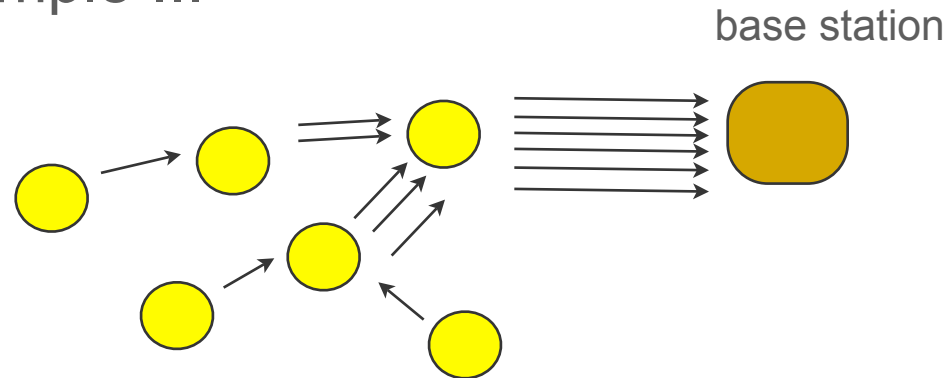
- In a sensor network, data can be stored:
 - **Externally**: edge of the sensor network
 - **Locally**: at the sensor node generating data
 - **In-network**: in a distributed way in intermediate nodes



[External Storage]

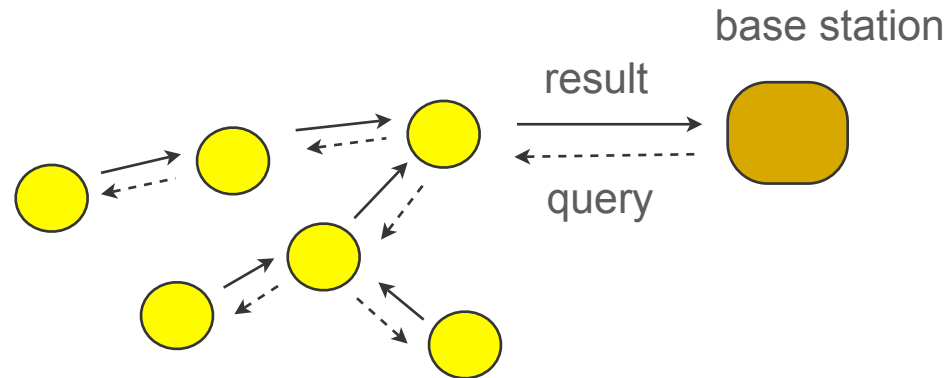
- Data are collected to an external sink as soon as they are collected

for each sample ...



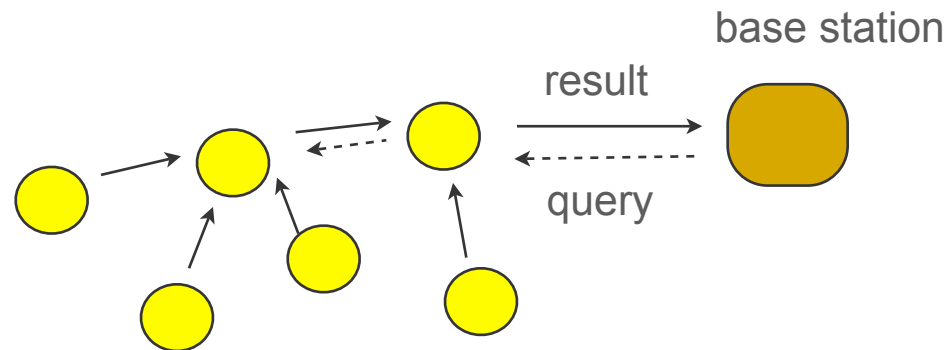
[Locally]

- Data stored locally in each sensor node
- Queries for data have to be flooded in the network
 - Direct Diffusion
 - TAG



In-network: Data-centric storage

- Data stored in particular nodes in the network
- Queries need to be addressed only to the nodes that store data
 - How to partition data? how to choose nodes?



TAG: a Tiny Aggregation Service for Ad-Hoc Sensor Networks

Samuel Madden, Michael J.
Franklin, Joseph Hellerstein
and Wei Hong

[Local Storage]

- Sampled data are stored on the sensor nodes
- A base station wants to query the data

*what is the maximum
air pressure?*

*what is the average
temperature in the
area?*

*which are the rooms
with average noise
level $> n$?*

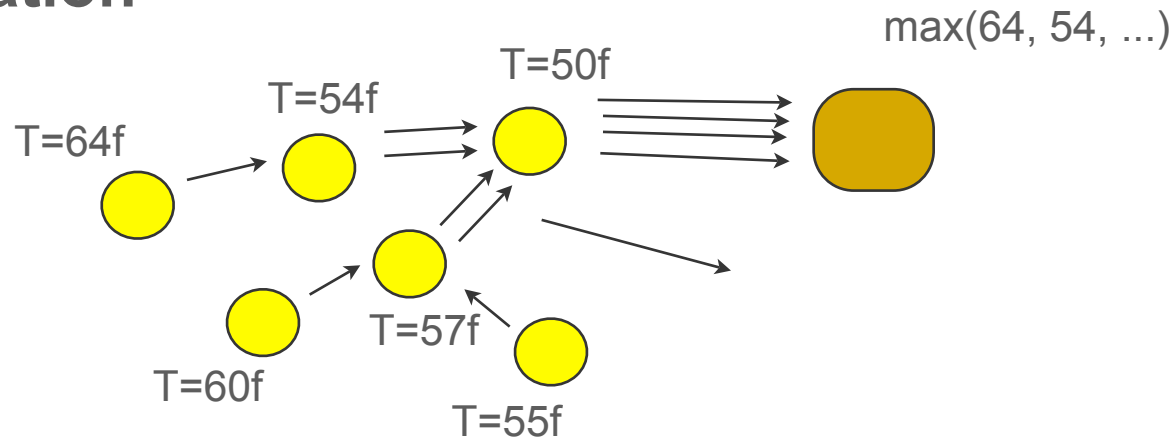
Aggregation is central in sensor network



[Motivation]

- Previous Systems
 - Data samples moved to an external base station and then analyzed

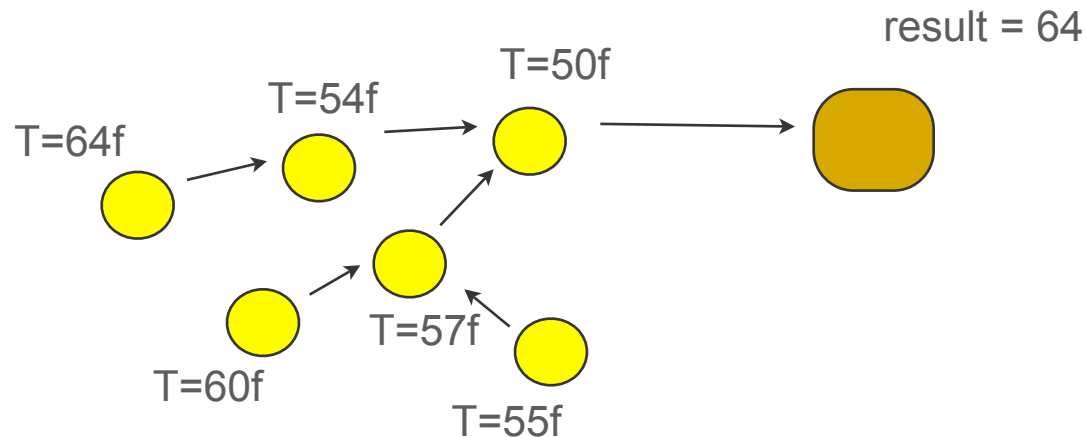
No aggregation



[Approach]

- TAG:
 - Push aggregation to the nodes to reduce transmission

max computed in the network



Structure of the Queries

- SQL-like queries

aggregation operator

SELECT avg(volume), room FROM sensors

WHERE floor = 6 ← filter on the sensors to consider

GROUP BY room ← partitioning of sensor readings

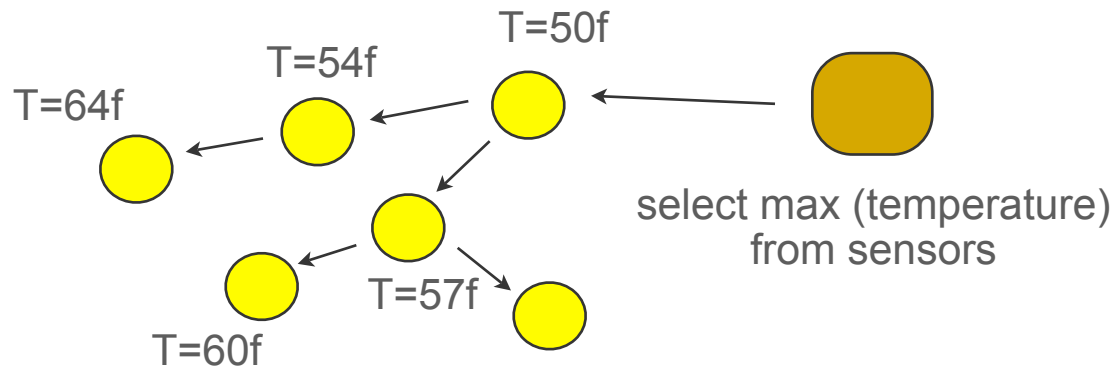
HAVING AVG(volume) > threshold ← filter on aggregated value

EPOCH DURATION 30s ← sampling rate



Distribution Phase

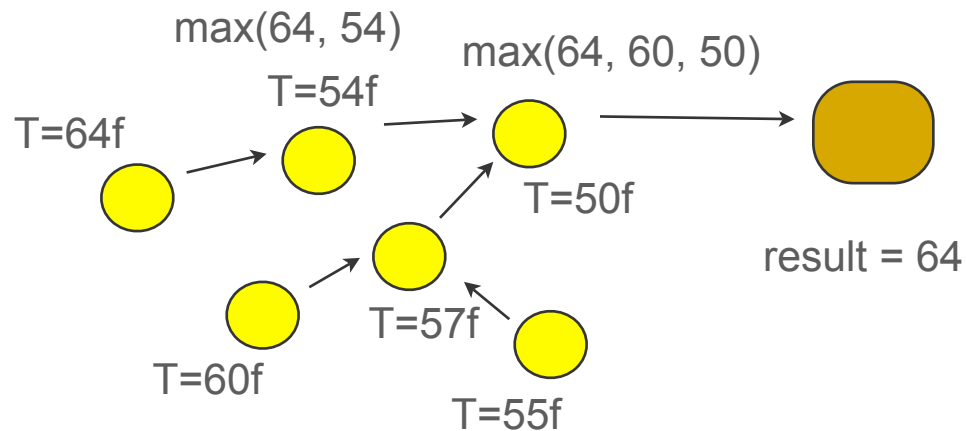
- Query is sent in the network looking for nodes that satisfy the filter conditions



- Each node synchronizes its timing information from the query
- Sets as “parent” the node from which it received the query
- When forwards the query specifies the interval in which it wants to receive the response from the children

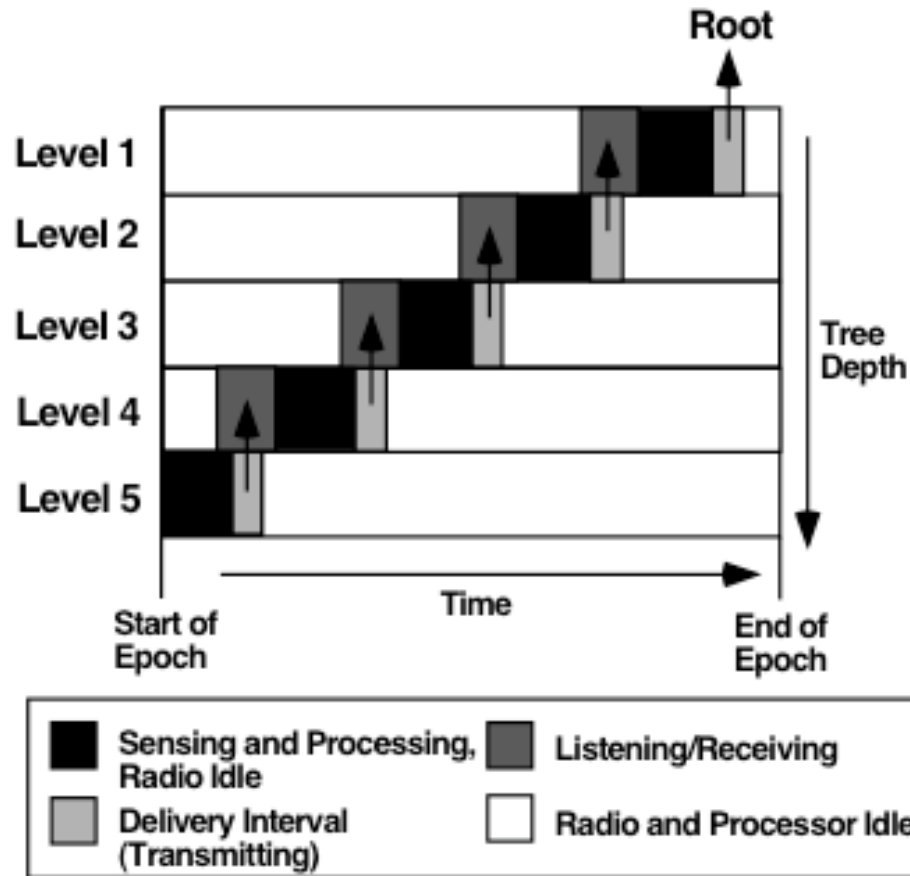
Collection Phase

- Results are collected and sent back to the base station



- Response is given during the assigned interval
- Nodes respond after receiving all the responses from children
- Each node computes a partial aggregate on the results that it receives

[Graphically ...]



Aggregates are not equals

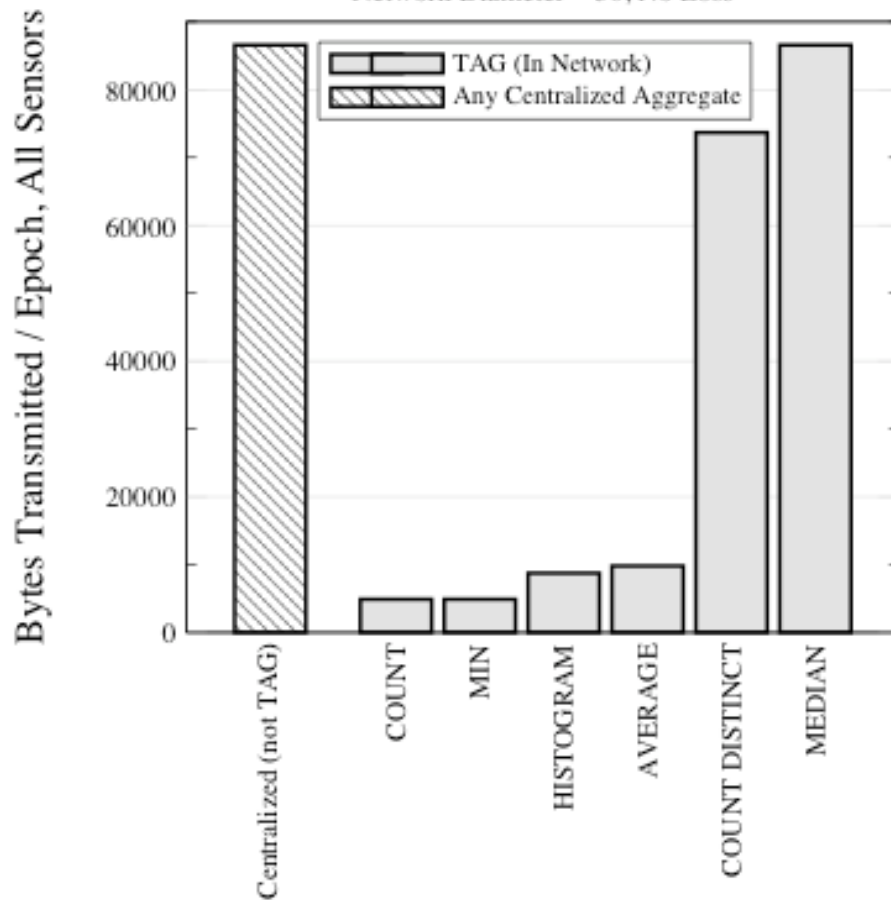
- Duplicate Sensitivity (count = Y, max = N)
- Exemplary / Summary (count = S, max = E)
- Monotonic (count = Y, max = Y, average = N)
- Partial State
 - Distributive: aggregate of the partition (count, max ...)
 - Algebraic: constant size (average)
 - Holistic: proportional to the size of the partition (median)
 - Unique: proportional to the # of distinct values (distinct)
 - Context-Sensitive: proportional to some property of the data (histograms)



Experimental Results

In-network vs. Centralized Aggregation

Network Diameter = 50, No Loss



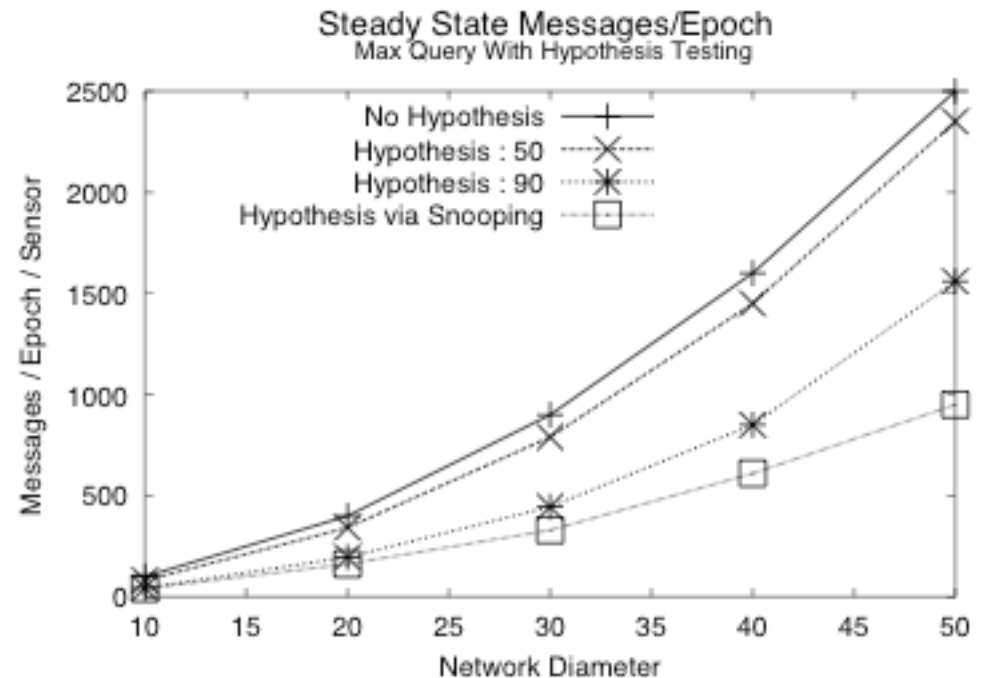
2500 nodes ($d=50$)

- Advantage of in-network aggregation depends on the type of aggregate to compute (holistic aggregates are less effective)
- No query propagation cost

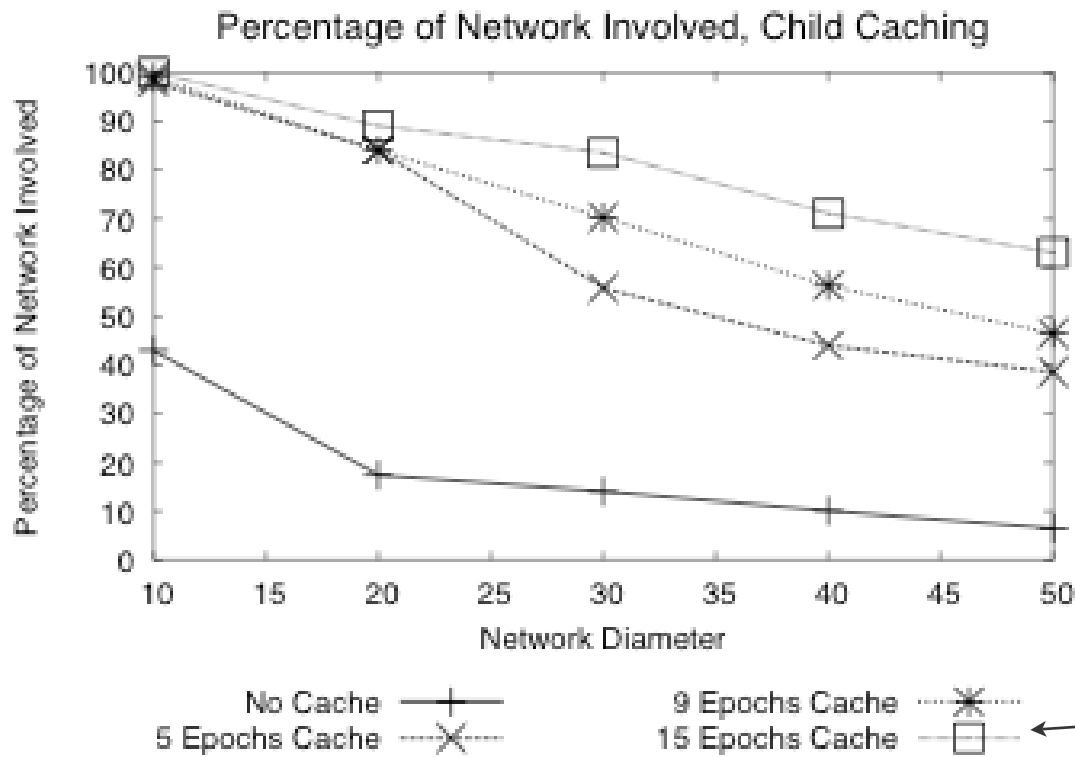
Grid Topology: if all in one hop no advantages in aggregating

Optimizations

- *Snooping approach*
 - Recover lost query message
 - Suppress sending values if sending it do not change the total value (max, min ...)
- *Hypothesis testing*
 - Compute an exemplary aggregate (min, max) over only m nodes
 - Issue a new query that searches for nodes with a better value



Effect of Lost Messages



Rate of lost packets determined from a realistic scenario

Nodes of the network that are involved in the aggregation after the faulty nodes die

Reply previous results from a children cache if child node dies



Discussion

- **Contribution:** Aggregation as core service / In-Network data aggregation / Dynamic tree construction / Real-time scheduling of responses
- Are persistent queries the most suitable approach for wireless sensor network?
 - Event Driven Queries?
- Should we use TAG over a reliable transport layer or should TAG be loss resistant?
 - Pros and cons
- If, for losses, results are computed on a partial network how do the root know that this result is not complete?



A decorative yellow circle is positioned behind the title text. A large black left square bracket and a yellow right square bracket are placed around the title text.

DIFS: A Distributed Index for Features in Sensor Networks

Authors

Greenstein et al.

USC, Intel Research, Berkeley



Contributions

- **In-network** storage of data
 - Partitioning of data based on event, location, range
 - Selection of nodes based on geographic hash
- Load balance
 - Generated traffic
 - Storage
- Fast and communication efficient searches
 - by Name
 - by Value range
 - by Location
- Allows to efficient answers to queries on the distribution of data

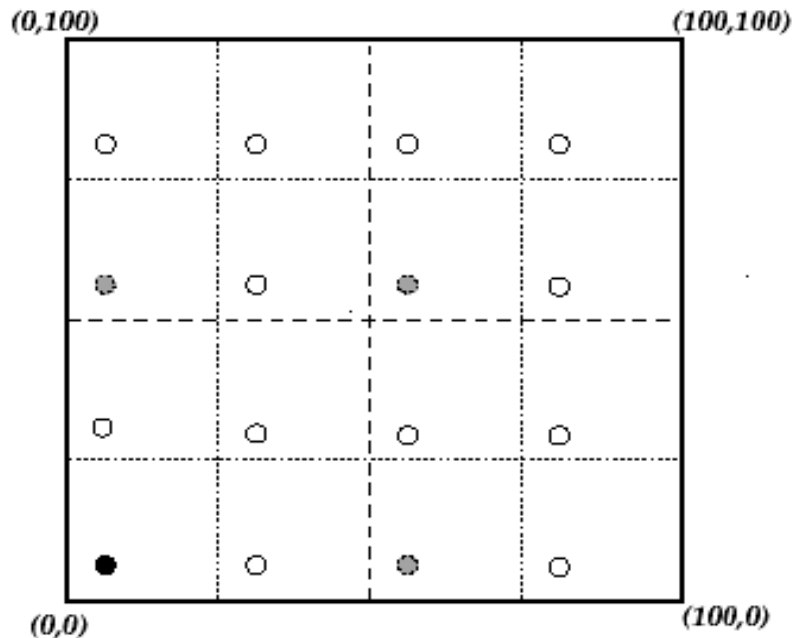


[Operations]

- Node
 - Generate data
 - Process data into High-Level events
 - Insert data into indices
- Users
 - Query Indices
 - Fetch data from:
 - Indices
 - Storage Nodes
 - Generating Nodes (for incomplete events)



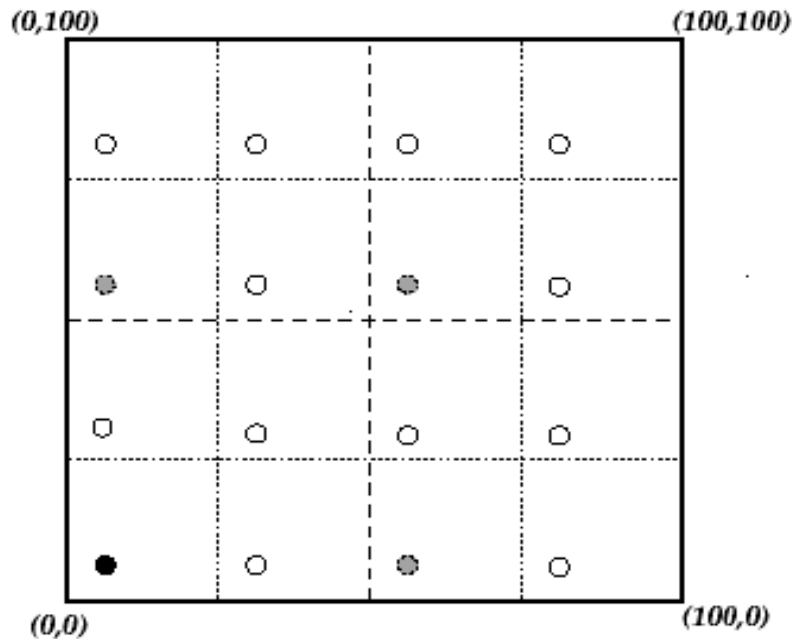
Approach I: GHT and Structured Replication



- *root point: (3,3)*
- *level 1 mirror points: (53,3) (3,53) (53,53)*
- *level 2 mirror points: (28,3) (3,28) (28,28) (78,3) (53,28) (78,28)*
(3,78) (28,53) (28,78) (78,53) (53,78)(78,78)

- **Keys:**
 - Event Names
 - Hashed to geographic position
- **Structured Replication to avoid Hotspots:**
 - Make $4d-1$ images.
 - Store closest mirror.
 - Query to all mirror nodes.

Approach I: GHT and Structured Replication



Keys:

- Event Names
- Hashed to geographic position

Structured Replication to avoid Hotspots:

- Make 4^{d-1} images.
- Store closest mirror.
- Query to all mirror nodes.

- root point
- level 1 mirror
- level 2 mirror

Range queries not efficient: need to visit all nodes

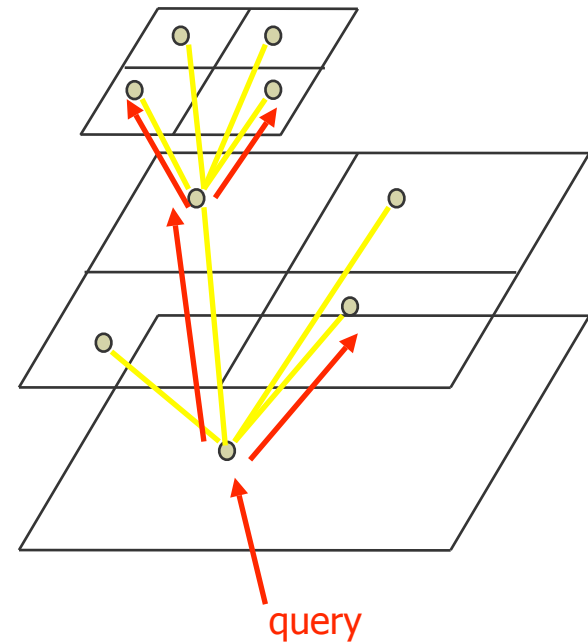
[Approach II: Quad tree]

Data stored locally

Each node maintains four histograms, one per quadrant

Search tree

Allows branch pruning



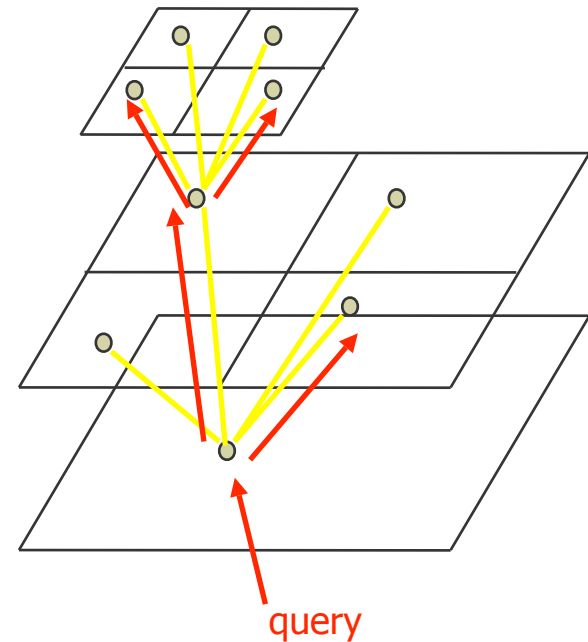
Approach II: Quad tree

Data stored locally

Each node maintains four histograms, one per quadrant

Search tree

Allows branch pruning



- Event notice must be propagated to the root
- Queries Must originate with the root

[DIFS: Let's go hybrid]

- Extends GHT and Quad tree:
 - Support efficient range queries
 - Maintains balanced load across nodes
- Creates multiple rooted hierarchical index
 - Non-root nodes can have multiple parents ($b_{fact} = 2^i$)
- Each node covers a space interval and a range:
 - The larger the space interval, the smaller the range



[DIFS design]

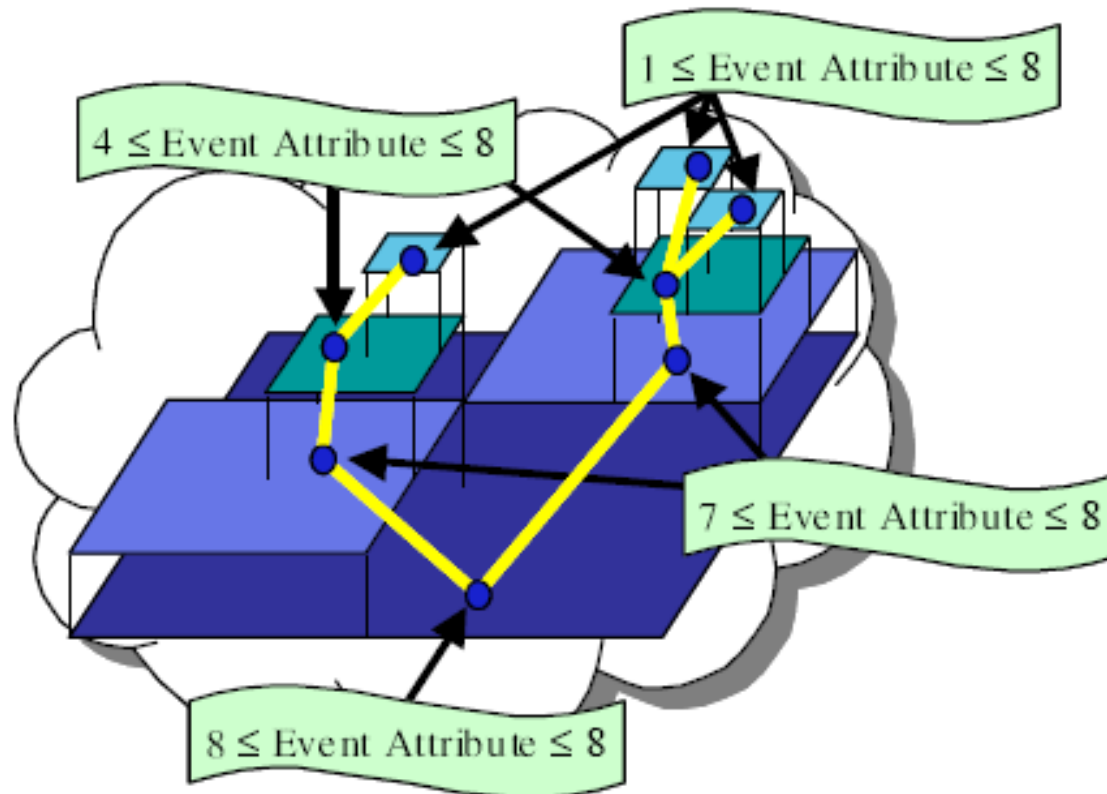


Fig. 3. An illustration of the DIFS hierarchy.

Example: Data Insertion

- Sensed value: 57
- Coordinates of event: (14,37)
- Minimum bounding box: 8 x 8 units
- Max Range: 0 - 255
- bfact: 2

	Spatial int.	Range
Local leaf	(8,32)(15,39)	0-255
Level 1	(0,32)(15,47)	0-127
Level 2	(0,32)(31,63)	0-63
Level 3	(0,0)(63,63)	0-31
	...	

[Queries]

- by Value
 - Value ranging from 47 to 68
 - bfact = 4
 - 1 level 4 node for value 47.
 - 1 level 2 nodes for range [48-63]
 - 1 level 3 nodes for range [64-67]
 - 1 level 4 node for value 68.
- over Space
 - handles location criteria specified by a bounding box.
 - query only to those index nodes that cover the any part of interest range
- by Distribution
 - “What is the minimum value in the top ten percent of values?”
 - Histograms contain a total count of values that exceeds the bounds



[Event remove]

- Explicit deletion
 - requires only removing a pointer and decrementing the histogram.
- Timeout deletion require time stamp.
 - Leaf index maintained timestamp with each source pointer.



[Simulations]

- 1024 meter wide square topology
- 2048 node
- Communication radius 25m
- Distribution for events
 - Uniform
 - Gradient: event value proportional to the x coordinates.
 - Hot spot: random 5 peak location.
- 2048 events over the time [0,10] at uniformly random location.

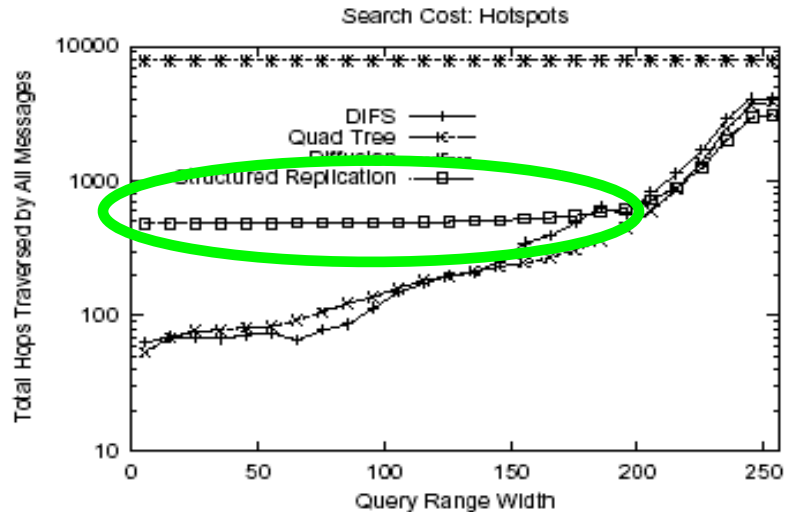
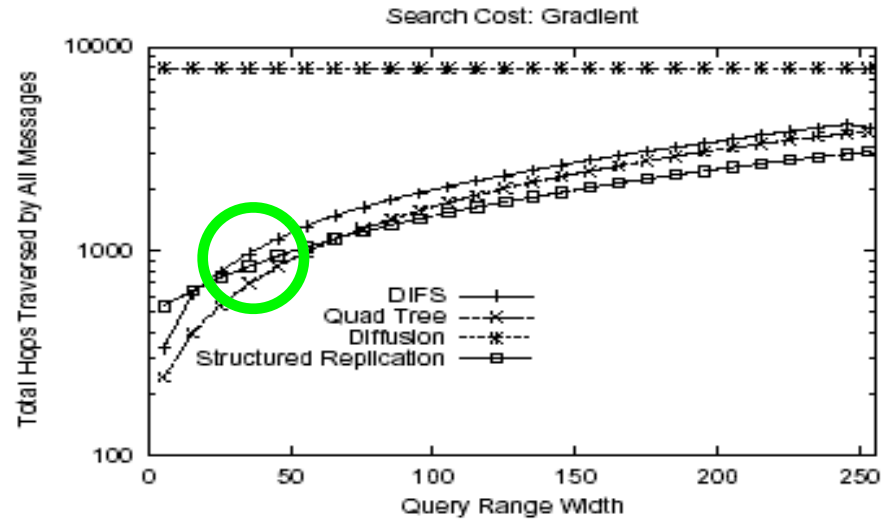
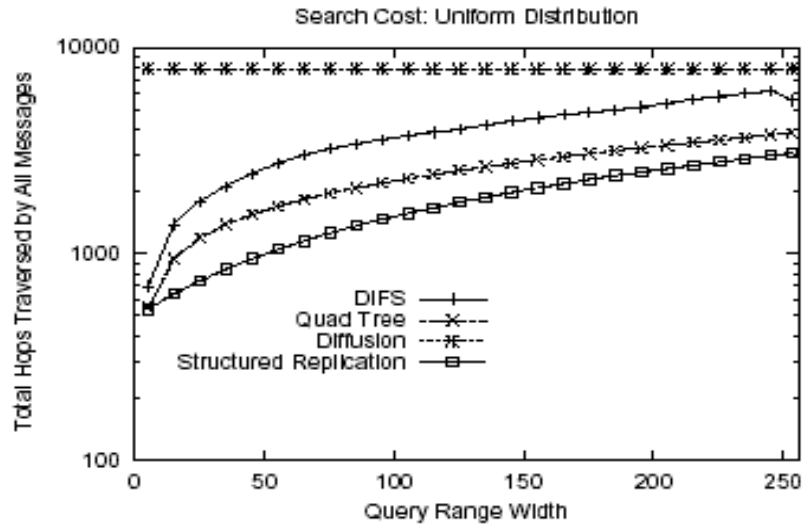
- Range queries simulated



[Simulations]

- 1024 meter wide square topology
- 2048 node
- Communication radius 25m
- Distribution for events
 - Uniform
 - Gradient: event value proportional to the x coordinates.
 - Hot spot: random 5 peak location.
- 2048 events over the time $[0,10]$ at uniformly random location.
- Range queries simulated

Search cost

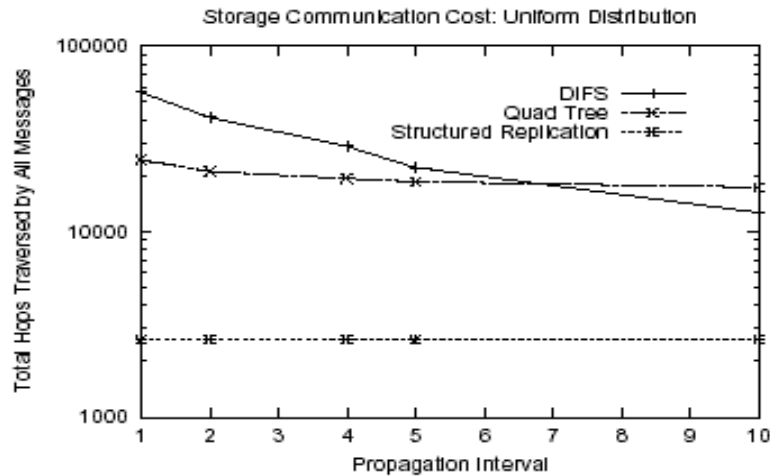


(b) Gradient

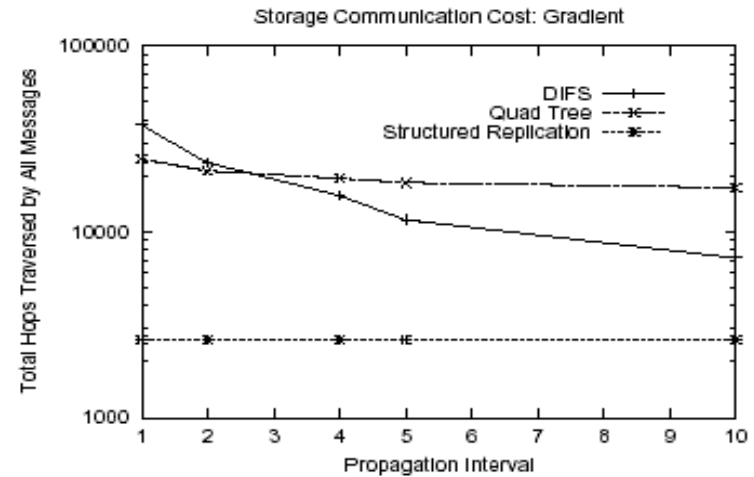
- GHT better in Uniform
- in non-uniform Quad Tree and DIFS better for narrow ranges
- In hotspots pruning of Quad tree and DIFS works!

(c) Hotspots

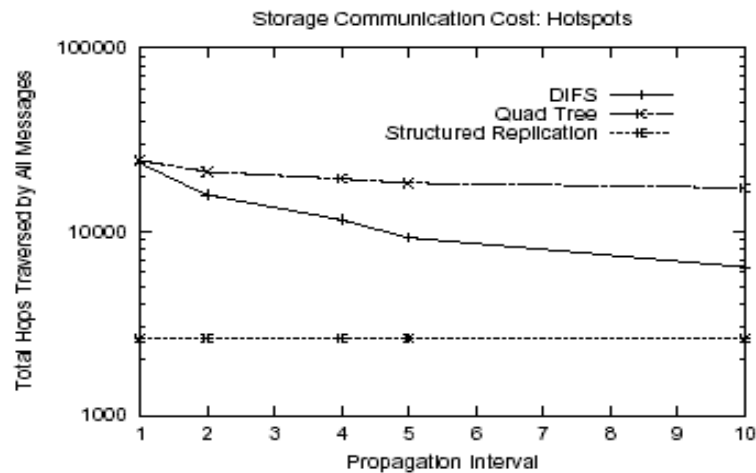
Storage Communication Cost



(a) Uniform Distribution



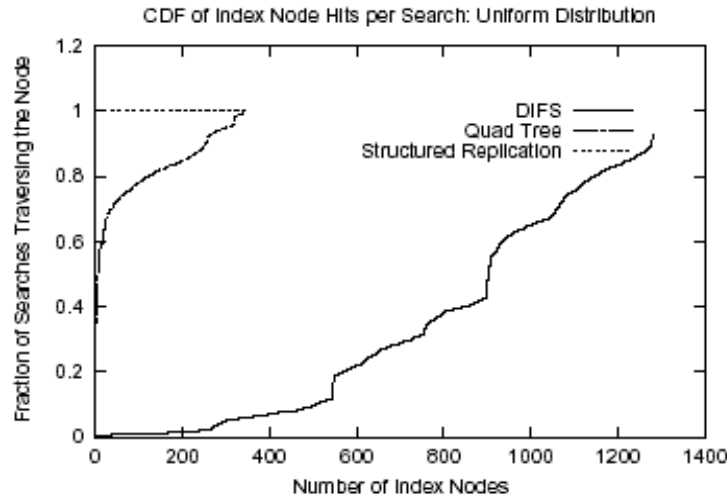
(b) Gradient



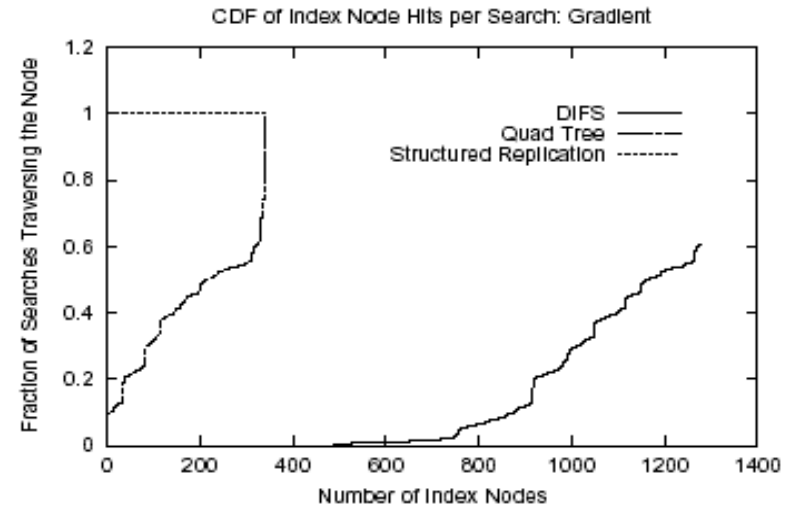
(c) Hotspots

- Structured replication performs always better
- DIFS is sensible to the propagation interval

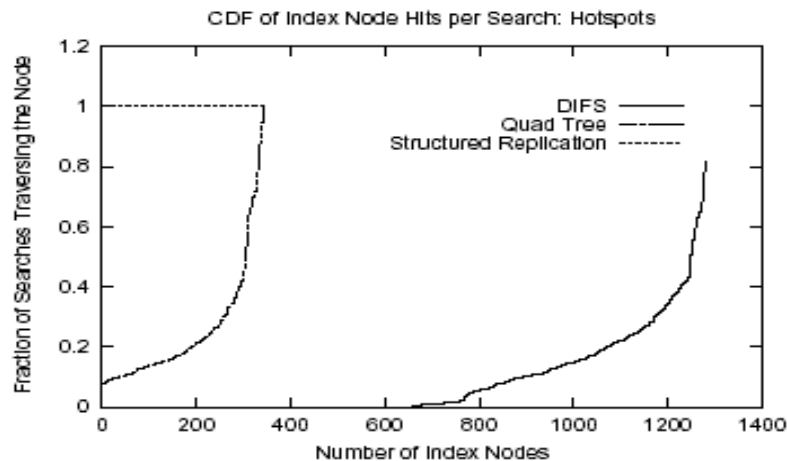
Results: Node hits per search



(a) Uniform Distribution



(b) Gradient



(c) Hotspots

- In Structured Replication all nodes are always visited
- in non-uniform DIFS performs significantly better

Discussion

- Managing of complex event
 - If an event has multiple attributes how is the system performing?
 - Complex queries on complex events: how many nodes do we need to visit to have an answer?
- This system works better with hotspot. Can we find a better way to explicitly deal with hotspots?
- Will the load be really balanced in a real sensing system (real range \ll largest possible range)
- A lot of decisions have to be made at design time (ranges, coordinate deformation to obtain uniform sensor distribution, propagation interval)
- For optimal values we need to know event distribution



A decorative yellow circle is positioned behind the title text. A large black left square bracket and a yellow right square bracket are placed around the title text.

A Framework for Time Indexing in Sensor Networks

Guanghui He, Rong Zheng,
Indranil Gupta, Lui Sha

[Overview]

- **Data-centric storage** method
- **Time** as attribute to partition data
 - Each piece of data (event, sample, ...) has a time associated to it.
 - Time is partitioned in intervals
- Data for each time interval is **stored on different nodes**



[Motivation]

- Time indexed queries are common

*how many cars crossed
the intersection at 10am on
Monday morning?*

*what is the average
temperature of last week?*

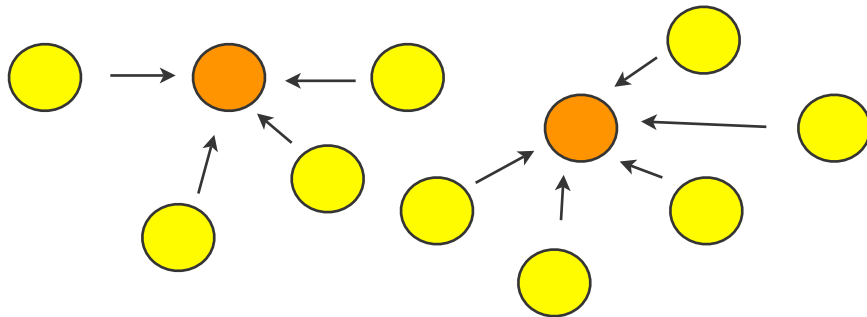
- Existing DCS:
 - expensive to send time-sequence data to far nodes



Proposed Solution: Overview

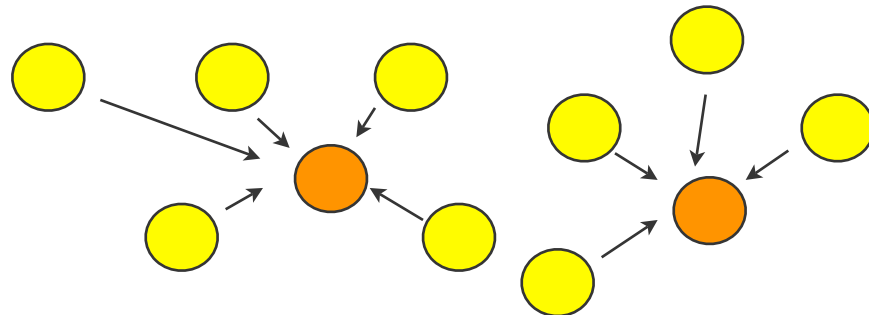
- Data is stored on Rendezvous Points (RPs)

t=2



- RPs form a *dominating set* of the network
- Each RP stores data for its neighbors

t=3

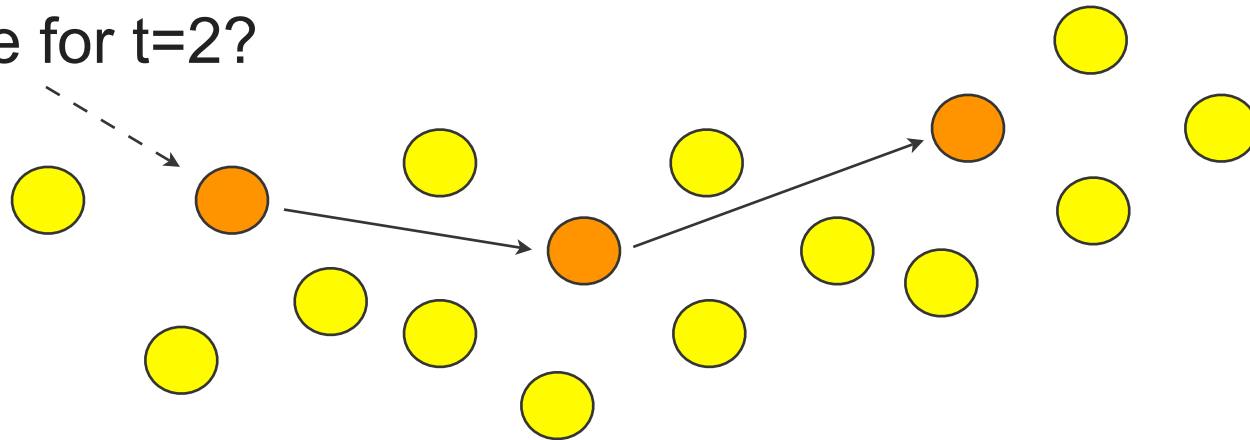


- Being RP is energy-consuming: RPs change at every time interval

[Query]

- RPs for each time interval connected in an overlay
- Queries for a specific time range goes to one of the overlays

temperature for $t=2$?



[Proposed Solution]

- RP role assignment view as a integer programming problem

$$\begin{aligned} \max \quad & \sum_{v \in V} U(k_v) \\ \text{s.t.} \quad & \sum_{u \in N(v)} s_u(i) \geq 1 \quad i = 1, 2, \dots, T \text{ and } \forall v \in V, \end{aligned}$$

Maximize an utility function

The solution needs to be a dominating set



[Which utility function?]

- Minimize energy?
 - x = number of times a node is RP \Rightarrow min x
 - Fixed minimal dominating set is the solution that minimize energy consumption
- Load balancing to increase the lifetime of the network
 - Logarithmic utility function:
 - $\max U(x) = \max \log(1-x/T)$
 - If a node is RP T times $\Rightarrow \log(T) = -\inf$



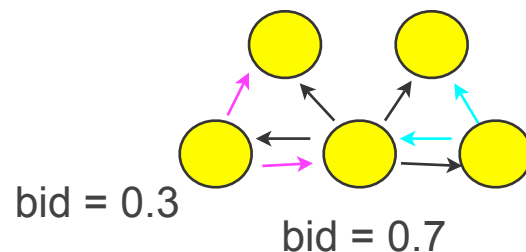
Solution of the Optimization

- Consider a relaxation of the Non-Linear Integer Programming
 - Variables can be **fractional**
 - Relaxation of the dominating-set constraint using the Lagrange multipliers method
 - Implemented as a distributed algorithm
- The solutions a_v of this problem can be interpreted as the probability that a node is an RP in each time interval
 - Used as input for an heuristic for the solution of the original optimization problem

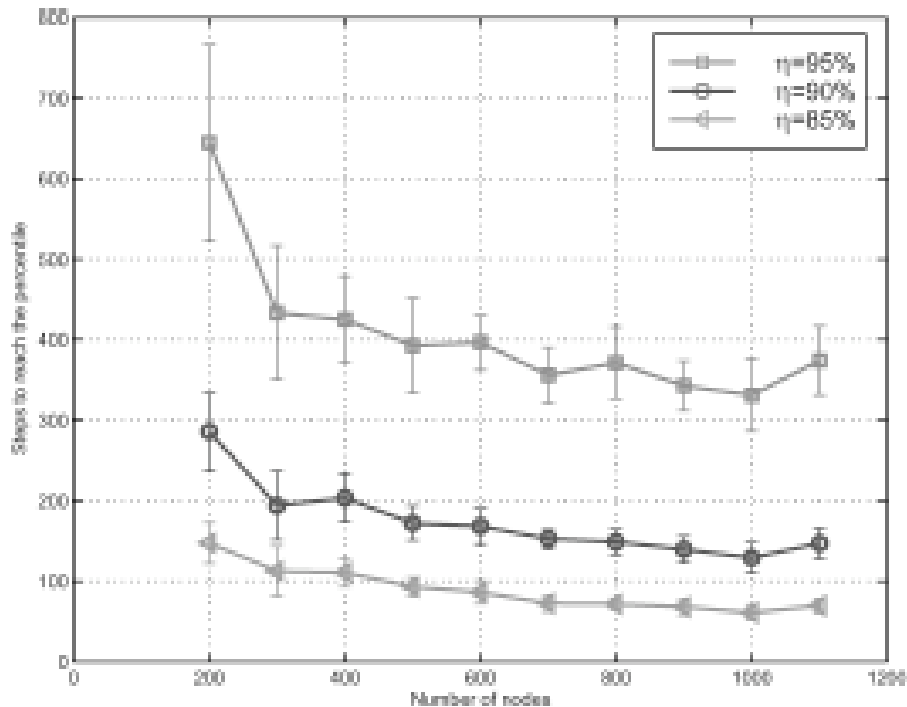


Heuristic Method

1. Each node creates a bid “proportional” to the a_v and sends it to its neighbors
2. If the node has the highest bid amount all or at least l of its neighbor, then the node becomes RP
3. Each node sends its decision to its neighbor.
4. In no node decided to be RP, then it elects itself as RP.

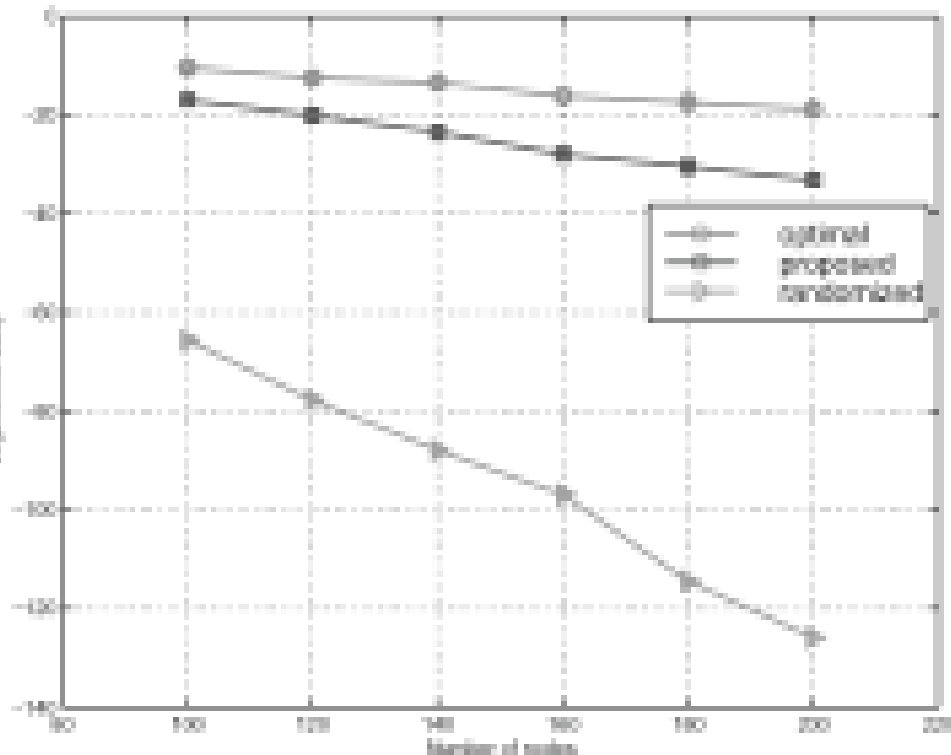


of Iterations



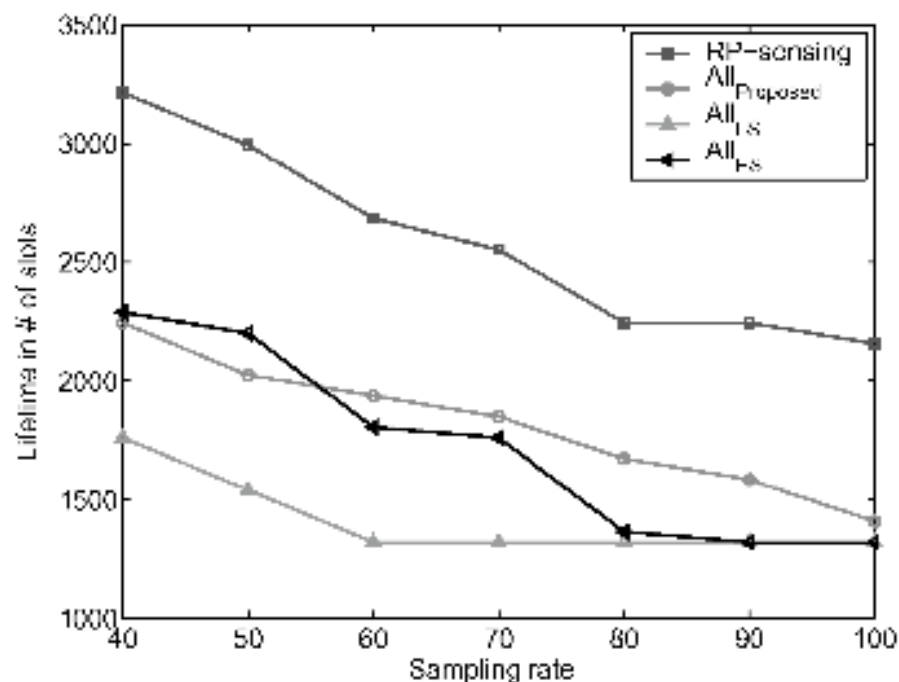
- Number of iteration needed to compute the solution with the relaxed bounds
- Random node position and fixed density

Heuristic Quality

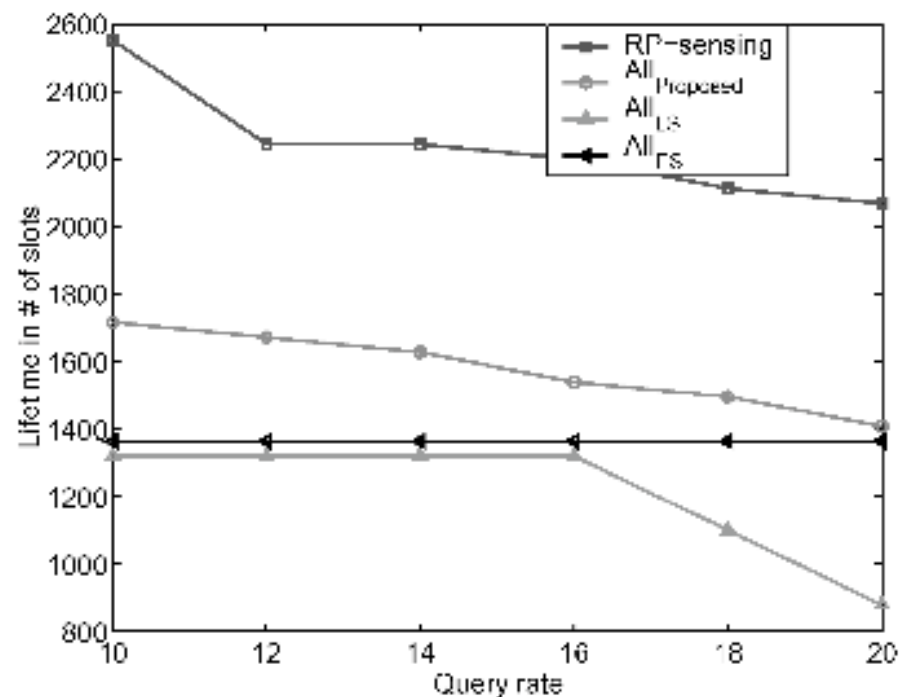


- RP election quality, measured using the proposed utility function
- Useful for understanding how far is the heuristic from the optimal value
- Randomized line: each node decides to be RP tossing a coin

Network Lifetime



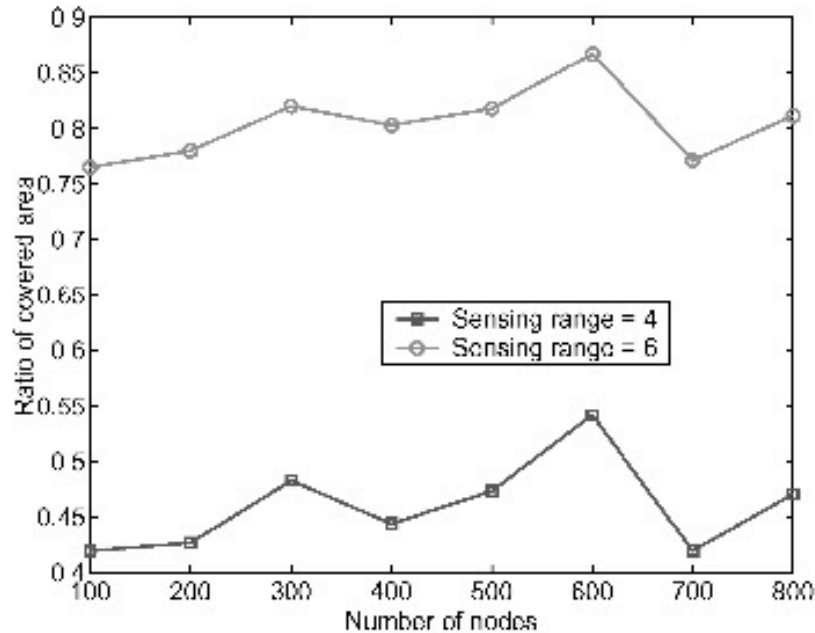
(a) Fixed query rate 14



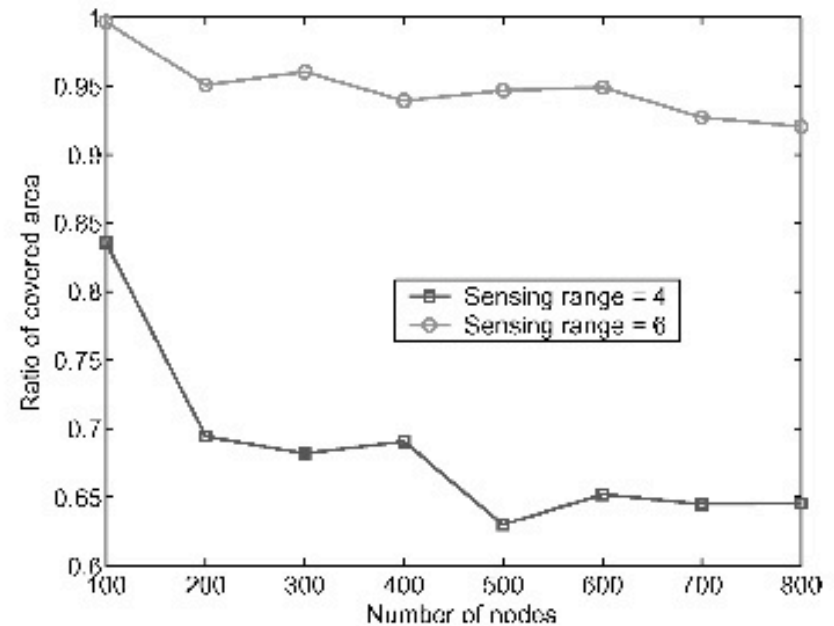
(b) Fixed sampling rate 80



Sensing modes



(a) Proposed algorithm



(b) Randomized algorithm

- Increasing in network lifetime can be obtained if, for each time interval, only the RP nodes perform sensing (the others can sleep)

Discussion

- Design of sensor networks: should we store data on sensors, externally or in-network? How do we decide?
- Is the effort in finding the optimal solution worth the gain in lifetime?
- How is the need for timing synchronization influence the network lifetime?
- If you use an adaptive sampling rate based on measurements, how is the network lifetime affected?
- What if the topology changes?



[Final Discussion]

- Local Storage approaches
 - TAG
- Data Centric Approaches
 - DIFS
 - Time Indexing

