

In-network Historical Data Storage and Query Processing Based on Distributed Indexing Techniques in Wireless Sensor Networks

Chunyu Ai¹, Ruiying Du^{1,2}, Minghong Zhang^{1,3}, and Yingshu Li¹

¹ Georgia State University, 34 Peachtree St., Atlanta, GA, USA

² Wuhan Computer School of Wuhan University, Wuhan, China

³ Department of Mathematics, Graduate University, Chinese Academy of Sciences, Beijing, China

WASA 2009, Boston.



Motivation

- Why do we need to store historical data?
- Where?
 - Base station
 - In sensor networks (sensor nodes)
- How to store and use historical data efficiently?

Outline

- Historical Data Storage
- Construct and Maintain Distributed Index Tree
- Historical Data Query Processing
- Simulation
- Conclusion

Historical Data Storage

- Is it possible to store historical sensing values on a sensor node locally?

Historical Data Storage

Mote	Flash memory	Measurements	Work load	Lifetime
Mica2	512 K bytes	100,000	Continuously running	5-6 days
Micaz	512 K bytes	100,000	Sleep	20 years
TelosB	1024 K bytes	200,000	Average	Several weeks or months

- Assume a Mica mote with a 512K bytes flash memory can live 3 months.
 $(100,000/90) = 1111$ measurements can be saved locally every day. Suppose we have 4 sensing attributes need to be saved, frequency of sampling can be $1111/(24*12*4)=1$ per 5 minutes which is normal in wireless sensor network applications.

Historical Data Storage

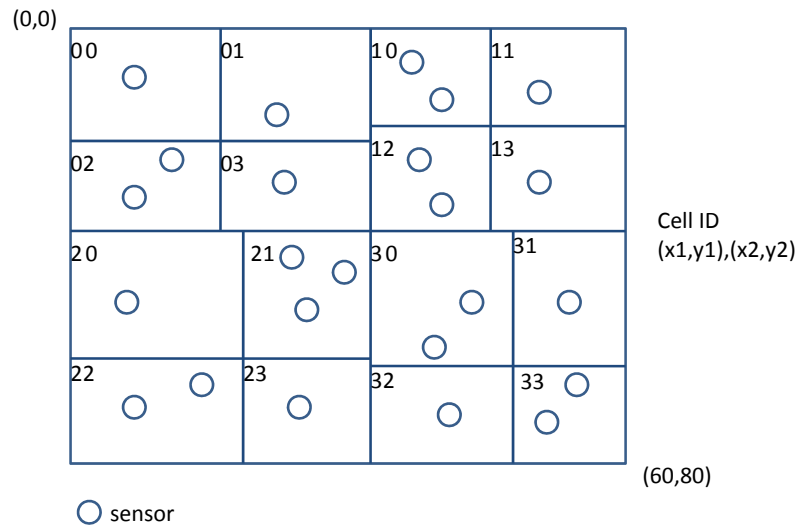
- There are n attributes (A_1, A_2, \dots, A_n) need to be recorded. For node N_i at time T_j , the sensing values are (V_1, V_2, \dots, V_n) respectively.

Then, $(T_j, V_1, V_2, \dots, V_n)$ is inserted into the flash memory.

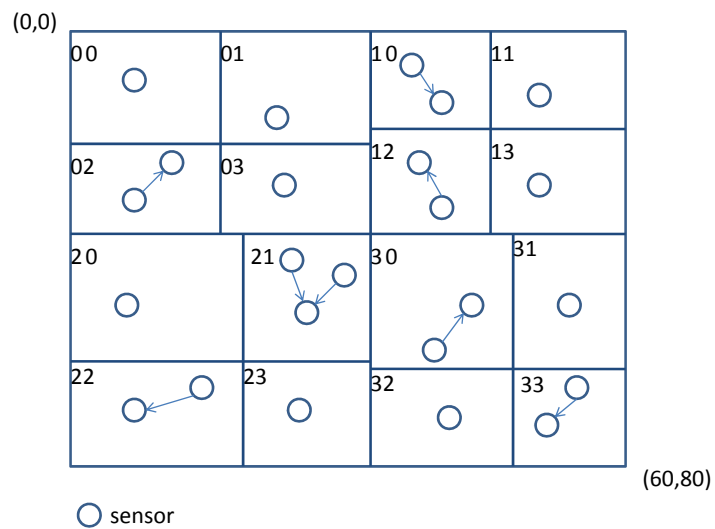
- When the flash memory is occupied more than 80%, a compression process is triggered.

$$T_1, T_2, T_3, T_4, T_5, \dots, T_k \rightarrow T_1, T_3, T_5, \dots, T_{\lfloor k/2 \rfloor}$$

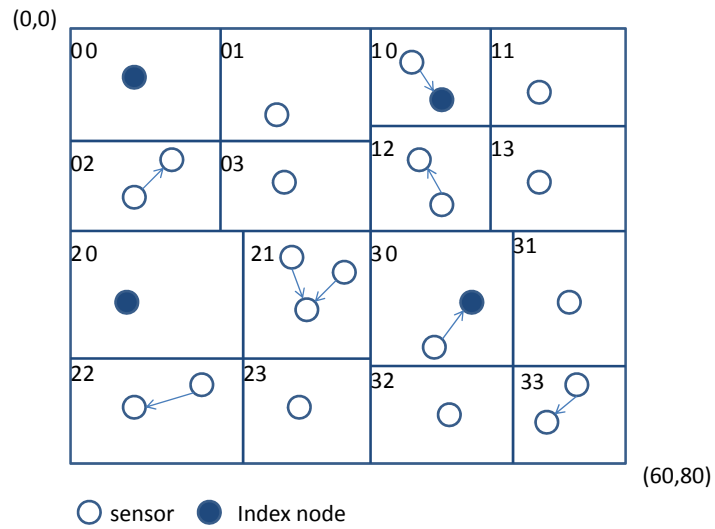
Construct Distributed Index Tree



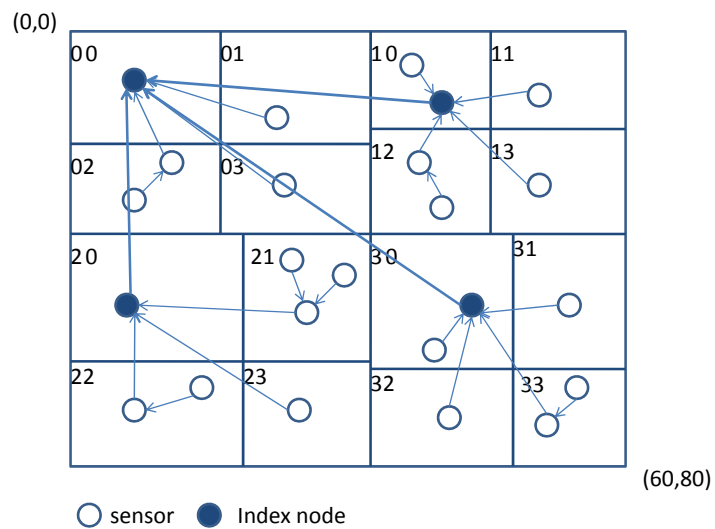
Construct Distributed Index Tree



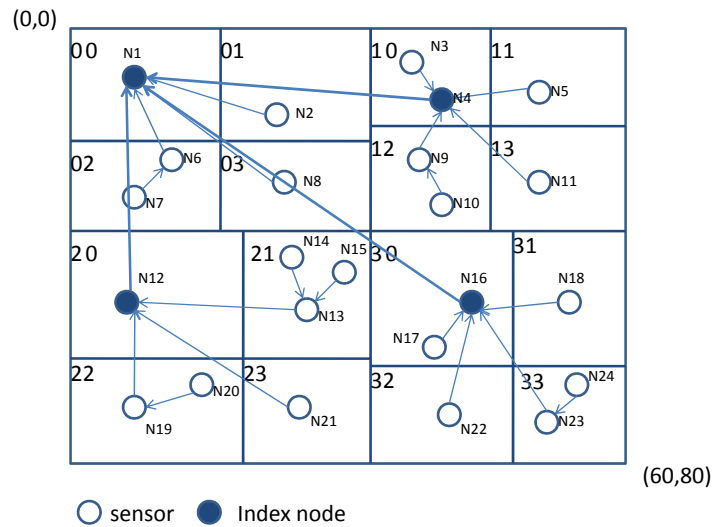
Construct Distributed Index Tree



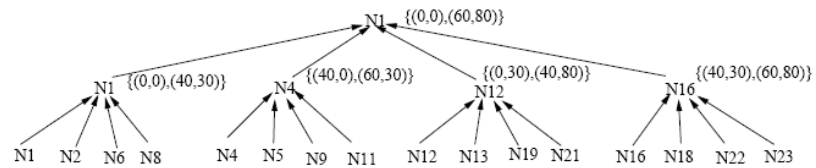
Construct Distributed Index Tree



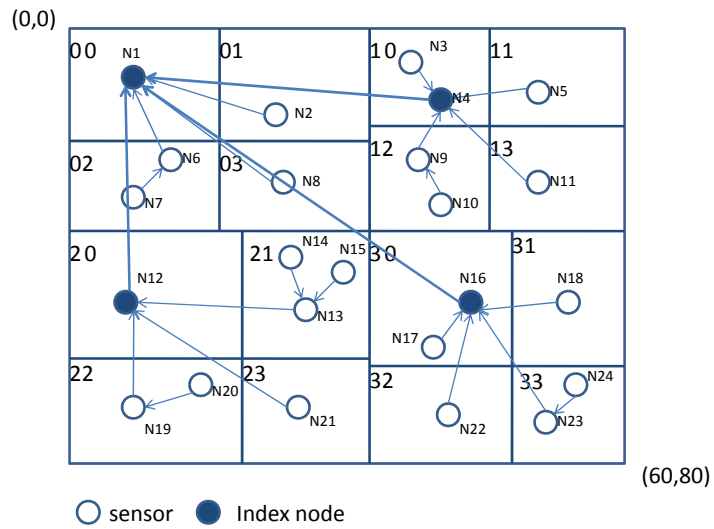
Construct Distributed Index Tree



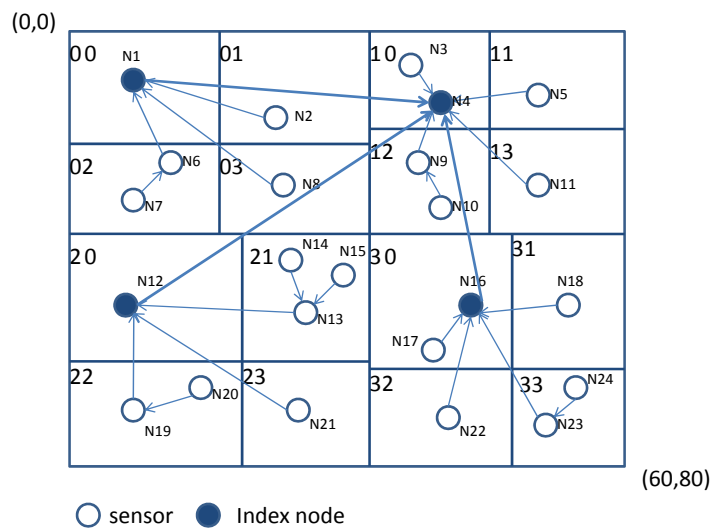
Construct Distributed Index Tree



Maintain Distributed Index Tree

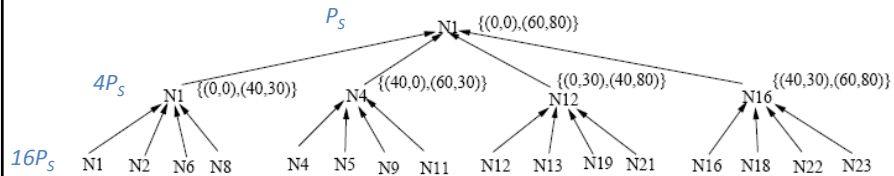


Maintain Distributed Index Tree



Maintain Distributed Index Tree

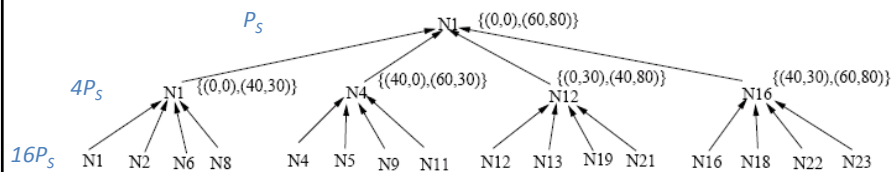
The administrator needs to specify a *tree switching period* P_s .



The switching period for index nodes in the layer L is $4^{L-1} P_s$

The ideal value of P_s is the estimated network lifetime divided by the number of sensor nodes.

Maintain Distributed Index Tree



For a node in the (L) th layer of an index tree, its parent's CID at time T can be calculated by replacing its own CID's $(L-1)$ th bit with $(\lfloor T / 4^{L-1} * P_s \rfloor \% 4)$.

Geographic routing is used to generate routes among nodes of the index tree since the CID of a sensor node also indicates its relative position.

Index Structure

- Each sensor node needs to calculate the maximum, minimum, and average for each attribute periodically. The administrator specifies an **update interval** which is much greater than the sensing interval.
- At the end of each update interval, a sensor node sends the index update message (**max, min, avg**) of that interval to its parent node of the index tree.

Index Structure

Child0:	Child1:	Child2:	Child3:
Location: $\{(x_{tl}, y_{tl}), (x_{br}, y_{br})\}$	Location: $\{(x_{tl}, y_{tl}), (x_{br}, y_{br})\}$
Attribute1: $T_i, \text{max, min, avg}$ $T_{i+1}, \text{max, min, avg}$	Attribute1: $T_i, \text{max, min, avg}$ $T_{i+1}, \text{max, min, avg}$		
⋮	⋮		
Attribute2: $T_i, \text{max, min, avg}$	Attribute2: $T_i, \text{max, min, avg}$		
⋮	⋮		
⋮	⋮		
⋮	⋮		

When an update message is received, it is inserted into the index structure. The internal node also merges four update messages for the current time interval T by calculating max, min, and avg, and sends this update message to its parent.

Historical Data Query Processing

- Historical data queries can inquire max, min, and avg values for a past period of time in a specified geographic area. Also, the sensing values of specific sensor nodes or locations in the past can be retrieved. A historical query can be issued at the base station or a sensor node in the network which is named as *query node*.

Query Example1:

```
SELECT S.temperature, S.humidity, S.location FROM sensor S
WHERE S.location WITHIN {(10, 10), (30, 30)} AND S.time BETWEEN now()-1 days AND now()
```

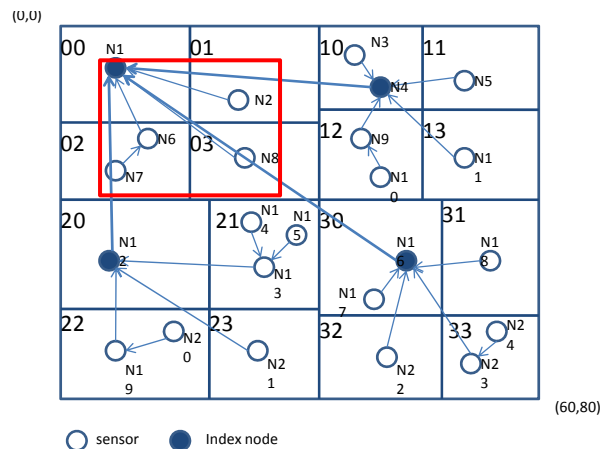
Query Example2:

```
SELECT MAX(S.temperature), S.location FROM sensor S
WHERE S.location WITHIN {(0, 0), (50, 30)} AND S.time BETWEEN 02/01/2009 and /02/04/2009
```

Historical Data Query Processing

Query Example1:

```
SELECT S.temperature, S.humidity, S.location FROM sensor S
WHERE S.location WITHIN {(10, 10), (30, 30)} AND S.time BETWEEN now()-1 days AND now()
```

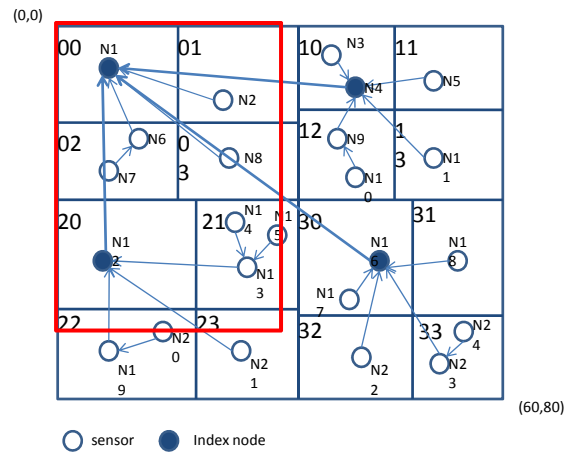


Historical Data Query Processing

Query Example2:

SELECT MAX(S.temperature), S.location FROM sensor S

WHERE S.location WITHIN {(0, 0), (50, 30)} AND S.time BETWEEN 02/01/2009 and /02/04/2009



Simulation Results

The sensing interval of a sensor is 5 minutes, and the update interval P_s of the index tree is 2 hours. Since network traffic greatly affects energy efficiency, we use it as the metric for performance evaluation.

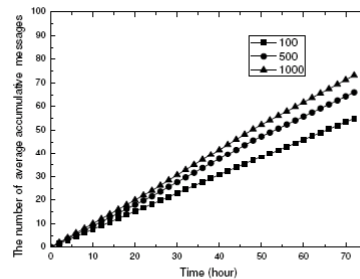
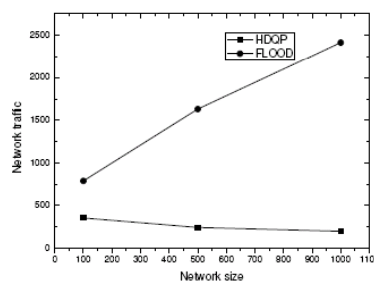
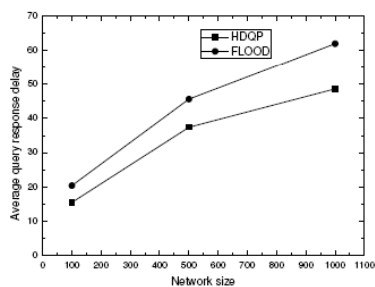


Fig. 3. Cost of maintaining the index tree with various network size.

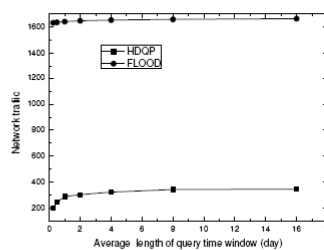
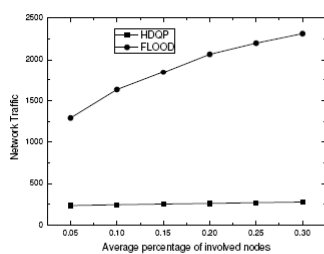
Simulation Results



(a) Query responding delay of variant network size. (b) Average Network traffic of variant network size.

Fig. 4. Query Processing Performance, 1000 queries during 72 hours.

Simulation Results



(c) Average Network traffic of variant average involved nodes percentage of queries. (d) Average Network traffic of variant average length of query time window.

Fig. 4. Query Processing Performance, 1000 queries during 72 hours.

Conclusion

- In this paper, we have proposed a scheme, HDQP, to process historical data queries of wireless sensor networks.
- Our approach is the first work to study distributed historical data query processing by using an effective distributed index tree.
- The indexes can help process queries energy-efficiently and reduce the query responding delay.
- Index tree switching mechanism also can balance the load among sensor nodes to avoid data distribution and energy consumption skew.

Thank you.