“Data Storage In Wireless Sensor Databases”

Demetris Zeinalipour
Department of Computer Science
University of Cyprus*

eNEXT WG1 Workshop on Sensor and Ad-hoc Networks
University of Cyprus - UCY, Nicosia, 13-14 March 2006

* Presented work was conducted at the University of California – Riverside.

http://www2.cs.ucy.ac.cy/~dzeina/
Presentation Goals

• To present a new **perspective** on **data management** and **query processing** related issues in sensor networks.

• This is an **overview talk** of various individual aspects that are important in this context.

• It does not focus on networking related technologies, but rather on how to organize the information generated by sensors in an energy-efficient manner.

D. Zeinalipour (UCY/CS)
Sensor Networks & The Silicon Era

• Applications:
  – Environmental and habitant monitoring
  – Seismic and Structural monitoring, ….

• Result:
  – Non-Intrusive/Non-Disruptive technology that enables the human to monitor and understand the physical world.

Wildlife Tracking: GPS Collars

Environmental Monitoring  Structural Monitoring
The typical SensorNet Framework

Sense and Send Paradigm

Sensors acquire environmental parameters and transmit these to the sink at pre-specified intervals

A Database that collects readings from many Sensors

Centralized:
• Storage, Indexing
• Query Processing
• Triggers, etc..

D. Zeinalipour (UCY/CS)
The typical SensorNet Framework

Data Acquisition

**TinyDB** (SIGMOD’03) and **Cougar** (CIDR’03) Frameworks:
- Provide a declarative SQL-like approach for accessing data.
- Are suitable for continuous queries.
- Push aggregation in the network (TAG - OSDI’02) but keep much of the processing at the sink.

\[
\text{SELECT \{AGG(expr), attrs\}} \\
\text{FROM \{table\}} \\
\text{WHERE \{selectPeds\}} \\
\text{GROUP BY attrs} \\
\text{HAVING \{havePeds\}} \\
\text{EPOCH DURATION \textit{i}}
\]

But Many applications do not require the query to be evaluated continuously…

D. Zeinalipour (UCY/CS)
Our Model: In-Situ Data Storage

1. Sensors acquire readings from their surrounding environment.
2. The data remains In-situ (at the generating site) in a sliding window fashion.
3. *When Users want to search/retrieve some information they perform on-demand queries.*

A network of Sensor Databases

- Distributed Storage
- Distributed Query Processing

Objective: To minimize the utilization of the radio

D. Zeinalipour (UCY/CS)
Our Motivation

• The **Bio-complexity** and **James Reserve** projects at **UC-Riverside**, where biologists want to utilize non-intrusive, not necessarily online, technologies to monitor **CO₂ levels in the soil**, rather than in laboratory recreations.

• Scientists do not need answers to their queries at all times.

• However a query execution has to adhere to the distinct characteristics of a Wireless Sensor Environment (minimize communication, local processing and aggregation, etc).
Challenges of the In-Situ Model

- **How to efficiently store information locally**
  
  **Solution:** We build the RISE Sensor that features an external flash memory – Giga-scale storage.
  
  [IEEE/ACM IPSN’05, IEEE SECON’05, ACM Senmetrics’05]

- **How to efficiently access a Giga-Scale storage medium of a Sensor Device?**
  
  **Solution:** We build the MicroHash Index Structure
  
  [IEEE NetDB (ICDE’05), USENIX FAST’05]

- **How to find the most important events without pulling together all distributed relations?**
  
  **Solution:** We build the Threshold Join Algorithm
  
  [IEEE DMSN’05 (VLDB’05)]
Talk Outline

1. The RISE Hardware Platform.

2. Indexing on Flash Memory of a Sensor Device.

3. Distributed top-k Query Processing.

4. Conclusions and Future Work.
A) The RISE Hardware Platform

The *RISE (Riverside SEnsor)* has been built as the prototype sensor platform demonstrating the In-Situ Data Storage Paradigm

- High performance, low power, state of the art platform
- Built around the **Chipcon CC1010** (System on Chip)
- Incorporates **TinyOS v1.1** (with nesC v1.2alpha1)
- Gigabyte scale - High capacity flash data storage (SD-Card)
- Multitude of sensors (Temperature, Carbon dioxide, Humidity, etc)
- Integrated radio transceiver - Compatible with MICA for interoperability and investigation into the nature of heterogeneous networks.
A) The RISE Hardware Platform

The RISE Sensor Specs

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Capability</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MCU</strong></td>
<td></td>
</tr>
<tr>
<td>Processor</td>
<td>24 MHz 8051 core</td>
</tr>
<tr>
<td>On-Chip Flash Memory</td>
<td>32 KB</td>
</tr>
<tr>
<td>Current (On, Idle, Off) at 14 MHz</td>
<td>14.8 mA, 8.2 mA, 0.2 μA</td>
</tr>
<tr>
<td><strong>Radio (RF Transceiver)</strong></td>
<td></td>
</tr>
<tr>
<td>Communication Rate</td>
<td>76.8 kbits/s</td>
</tr>
<tr>
<td>Communication Range</td>
<td>250m at 868/915 MHz</td>
</tr>
<tr>
<td>Current (Receive, Send at 10dBm)</td>
<td>11.9 mA, 26.6 mA</td>
</tr>
<tr>
<td><strong>SD Card &amp; SPI Bus</strong></td>
<td></td>
</tr>
<tr>
<td>SPI bus rate</td>
<td>Up to 3 Mbps</td>
</tr>
<tr>
<td>Data page size</td>
<td>512 bytes</td>
</tr>
<tr>
<td>Data block size</td>
<td>16 KBytes</td>
</tr>
<tr>
<td>Current (Read, Write, Delete)</td>
<td>1.17 mA, 37 mA, 57 μA</td>
</tr>
<tr>
<td>Time (Read, Write, Delete) (512B)</td>
<td>6.25 ms, 6.25 ms, 2.26 ms</td>
</tr>
</tbody>
</table>

Table 1. Characteristics of the RISE platform.

The RISE Storage board (RISE v2) [IEEE SECON’05, SenMetrics’05]

1. Data in RISE v1 is stored on the external SDMedia (NAND flash).
2. NAND flash is not suitable for accessing data at a byte granularity.
3. RISE Storage Board features NOR flash (efficient byte-level granularity) and the SDMedia Card.
4. It complements RISE v1

IPSN’05
Talk Outline

1. The RISE Hardware Platform.

2. Indexing on Flash Memory of a Sensor Device.

3. Distributed top-k Query Processing.

4. Conclusions and Future Work.
B) Indexing on Sensor Devices

- **Task:** “Find from local storage all records that satisfy some query predicate” (e.g. temp=95F)
- The most prevalent volatile medium for a Sensor Devices is **Flash Memory**.

**Flash (NAND) Advantages**
- Simple Cell Architecture
- Economical Reproduction
- Shock Resistant
- Fast Read Times
- Power Efficient

---

D. Zeinalipour (UCY/CS)
B) Indexing on Sensor Devices

Why is Flash so different from other Storage Mediums (disks, ram, etc)?

(1) Read-Constraint: Reading data stored on flash memory can be performed at granularity ranging from a single byte to a whole block (typically 8KB-64KB).

(2) Delete-Constraint: Deleting data stored on flash memory can only be performed at a block granularity (i.e. 8KB-64KB).

(3) Write-Constraint: Writing data can only be performed at a page granularity (typically 256B-512B), after the respective page (and its respective 8KB-64KB block) has been deleted.

(4) Wear-Constraint: Each page can only be written a limited number of times (typically 10,000-100,000).
B) Indexing on Sensor Devices

- There is no related work on Local Indexes for Sensor Device Databases (most research focuses on Magnetic Disk and Main Memory Databases)
- We developed the MicroHash Index [FAST’05] which is an efficient structure to this problem.
- We also developed efficient Search algorithms that locate information stored on flash.
- **Main Idea:** Minimize expensive random access deletions
B) Indexing on Sensor Devices

- We have implemented all these algorithms in nesC, the programming language of TinyOS.
- Extensive trace-driven simulations using 5-year long temperature/humidity datasets from the University of Washington.
- We also used datasets from the Great Duck Island Study in Maine (UC-Berkeley)
B) Indexing on Sensor Devices

- Finding a record **by a value** (e.g. temp=95F) can be performed in constant time.
- Finding a record **by timestamp** (e.g. 14/3/06 10:30:00) can be performed in 3-6 page reads.

**Great Duck Island Study**

<table>
<thead>
<tr>
<th>Index On Attribute</th>
<th>Overhead Ratio $\Phi$(%)</th>
<th>Energy Index (mJ)</th>
<th>ScaleSearch Avg Page Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>Light</td>
<td>26.47</td>
<td>4,134</td>
<td>4.45</td>
</tr>
<tr>
<td>Temperature</td>
<td>27.14</td>
<td>4,172</td>
<td>5.45</td>
</tr>
<tr>
<td>Thermopile</td>
<td>24.08</td>
<td>4,005</td>
<td>6.29</td>
</tr>
<tr>
<td>Thermistor</td>
<td>14.43</td>
<td>3,554</td>
<td>5.10</td>
</tr>
<tr>
<td>Humidity</td>
<td>7.604</td>
<td>3,292</td>
<td>2.97</td>
</tr>
<tr>
<td>Voltage</td>
<td>20.27</td>
<td>3,771</td>
<td>4.21</td>
</tr>
</tbody>
</table>

Note: Storing records without index 3042mJ

D. Zeinalipour (UCY/CS)
Talk Outline

1. The RISE Hardware Platform.

2. Indexing on Flash Memory of a Sensor Device.

3. Distributed top-k Query Processing.

4. Conclusions and Future Work.

D. Zeinalipour (UCY/CS)
3) Distributed Top-K Query Processing

Motivating Example (Problem Formulation)

- Assume that we have \( n=5 \) sensor each of maintains locally a sliding window of \( m=5 \) readings. (See table)
- **TOP-1 Query**: “On which timestamp did we have the highest temperature across all sensors?”
- Note: \( \text{Score}(o_i) \) can only be calculated if we combine the readings from all 5 sensor.

<table>
<thead>
<tr>
<th>List</th>
<th>Sensor(s)</th>
<th>Local score</th>
<th>Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>o3,99</td>
<td>o1,91</td>
<td>o1,92</td>
</tr>
<tr>
<td>s2</td>
<td>o1,66</td>
<td>o3,90</td>
<td>o3,75</td>
</tr>
<tr>
<td>s3</td>
<td>o0,63</td>
<td>o0,61</td>
<td>o4,70</td>
</tr>
<tr>
<td>s4</td>
<td>o2,48</td>
<td>o4,07</td>
<td>o2,16</td>
</tr>
<tr>
<td>s5</td>
<td>o4,44</td>
<td>o2,01</td>
<td>o0,01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>TOP-5</strong></th>
<th>Complete Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>o3,4.05/5 = .81</td>
<td></td>
</tr>
<tr>
<td>o1,3.63/5 = .73</td>
<td></td>
</tr>
<tr>
<td>o4,2.07/5 = .41</td>
<td></td>
</tr>
<tr>
<td>o0,1.88/5 = .32</td>
<td></td>
</tr>
<tr>
<td>o2,1.75/5 = .29</td>
<td></td>
</tr>
</tbody>
</table>
Current Approach: TAG

- Aggregate the lists before these are forwarded to the parent

- This is essentially the TAG approach (Madden et al. OSDI '02)

- **Advantage:** Only \((n-1)\) messages

- **Drawback:** Sending everything!

D. Zeinalipour (UCY/CS)
TJA Step 1 (LB Phase)

- Each node sends its top-k results to its parent.
- Each intermediate node performs a union of all received lists (denoted as $\tau$):

<table>
<thead>
<tr>
<th>$v_1$</th>
<th>$v_2$</th>
<th>$v_3$</th>
<th>$v_4$</th>
<th>$v_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>o3, 99</td>
<td>o1, 91</td>
<td>o1, 92</td>
<td>o3, 74</td>
<td>o3, 67</td>
</tr>
<tr>
<td>o1, 66</td>
<td>o3, 90</td>
<td>o3, 75</td>
<td>o1, 56</td>
<td>o4, 67</td>
</tr>
<tr>
<td>o0, 63</td>
<td>o0, 61</td>
<td>o4, 70</td>
<td>o2, 56</td>
<td>o1, 58</td>
</tr>
<tr>
<td>o2, 48</td>
<td>o4, 07</td>
<td>o2, 16</td>
<td>o2, 54</td>
<td>o1, 58</td>
</tr>
<tr>
<td>o4, 44</td>
<td>o2, 01</td>
<td>o0, 01</td>
<td>o4, 19</td>
<td>o0, 35</td>
</tr>
</tbody>
</table>

Query: TOP-1

LB

{03, 01}
TJA Step 1 (HJ Phase)

- Disseminate $\tau$ to all nodes
- Each node sends back everything with score above all objectIDs in $\tau$.
- Before sending the objects, each node tags as incomplete scores that could not be computed exactly (upper bound)

<table>
<thead>
<tr>
<th>$v_1$</th>
<th>$v_2$</th>
<th>$v_3$</th>
<th>$v_4$</th>
<th>$v_5$</th>
<th>$HJ$</th>
</tr>
</thead>
<tbody>
<tr>
<td>o3, 99</td>
<td>o1, 91</td>
<td>o1, 92</td>
<td>o3, 74</td>
<td>o3, 67</td>
<td>o3, 405</td>
</tr>
<tr>
<td>o1, 66</td>
<td>o3, 90</td>
<td>o3, 75</td>
<td>o1, 56</td>
<td>o4, 67</td>
<td>o1, 363</td>
</tr>
<tr>
<td>o0, 63</td>
<td>o0, 61</td>
<td>o4, 70</td>
<td>o2, 56</td>
<td>o1, 58</td>
<td>o1, 58</td>
</tr>
<tr>
<td>o2, 48</td>
<td>o4, 07</td>
<td>o2, 16</td>
<td>o0, 28</td>
<td>o2, 54</td>
<td>o4, 44</td>
</tr>
<tr>
<td>o4, 44</td>
<td>o2, 01</td>
<td>o0, 01</td>
<td>o4, 19</td>
<td>o0, 35</td>
<td>o4', 354</td>
</tr>
</tbody>
</table>

D. Zeinalipour (UCY/CS)
TJA Step 1 (CL Phase)

Have we found K objects with a complete score?

Yes: The answer has been found!

No: Find the complete score for each incomplete object (all in a single batch phase)

- CL ensures correctness!
- This phase is rarely required in practice.

<table>
<thead>
<tr>
<th></th>
<th>v1</th>
<th>v2</th>
<th>v3</th>
<th>v4</th>
<th>v5</th>
<th>TOP-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>o0, 63</td>
<td>o1, 66</td>
<td>o0, 61</td>
<td>o4, 07</td>
<td>o4, 01</td>
<td>o0, 35</td>
<td></td>
</tr>
<tr>
<td>o1, 66</td>
<td>o1, 91</td>
<td>o3, 90</td>
<td>o3, 75</td>
<td>o1, 56</td>
<td>o1, 67</td>
<td></td>
</tr>
<tr>
<td>o3, 99</td>
<td>o3, 92</td>
<td>o4, 70</td>
<td>o4, 70</td>
<td>o2, 54</td>
<td>o1, 363</td>
<td></td>
</tr>
<tr>
<td>o1, 44</td>
<td>o4, 75</td>
<td>o1, 28</td>
<td>o4, 19</td>
<td>o2, 58</td>
<td>o0, 188</td>
<td></td>
</tr>
<tr>
<td>o2, 48</td>
<td>o4, 91</td>
<td>o2, 16</td>
<td>o0, 28</td>
<td>o4, 207</td>
<td></td>
<td></td>
</tr>
<tr>
<td>o0, 63</td>
<td>o0, 61</td>
<td>o0, 01</td>
<td>o4, 19</td>
<td>o2, 175</td>
<td></td>
<td></td>
</tr>
<tr>
<td>o0, 63</td>
<td>o0, 61</td>
<td>o0, 01</td>
<td>o4, 19</td>
<td>o2, 175</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Talk Outline

1. The RISE Hardware Platform.

2. Indexing on Flash Memory of a Sensor Device.

3. Distributed top-k Query Processing.

4. Conclusions and Future Work.
Conclusions and Future Work

- In-Situ Data Storage is a new approach for **data management** in Sensor Networks

- We want to incorporate the presented ideas in a unified **In-Situ Storage and Retrieval Management System**, similar to TinyDB, but distributed.
Related Publications

• **Indexing on Flash Memory**
  - **MicroHash Index:**

  - Indexing Spatiotemporal Records MicroGF – Online Compression Algorithms:

• **TOP-K Query Processing & In-Situ Data Storage**


Related Publications (cont.)

• RISE Hardware platform


“Data Storage In Wireless Sensor Databases”

Demetris Zeinalipour
Department of Computer Science
University of Cyprus*

Thank You!

eNEXT WG1 Workshop on Sensor and Ad-hoc Networks
University of Cyprus - UCY, Nicosia, 13-14 March 2006

* Presented work was conducted at the University of California – Riverside.

http://www2.cs.ucy.ac.cy/~dzeina/