Developing and Deploying Sensor Network Applications with AnduIN

Daniel Klan
Katja Hose
Kai-Uwe Sattler

Databases & Information Systems Group

Ilmenau University of Technology, Germany
Outline

- Introduction
- AnduIN
- Query Specification
- Query Optimization
- Case Study
- Conclusion
Introduction

- Anomalies in sensor networks
- Sensors measuring the temperature
Introduction

- Anomalies in sensor networks
- Detecting anomalies in sensor data streams
Introduction

- Anomalies in sensor networks
- Determine anomaly regions
Introduction

- Wireless sensor networks
  - Continuous monitoring (environment, traffic, ...)
  - Limited capacities (computing power, battery lifetime, ...)
- Query processing
  - Centralized processing
  - In-network query processing
  - Hybrid processing
Introduction

Problem:
- Which parts of a query to evaluate within the network?

AnduIN:
- Combining in-network query processing with data stream processing
- Query decomposition and optimization, objective: minimization of energy consumption
- Comfortable way of specifying queries
Logical Sensor Networks

- Registering a sensor
  
  ADD SENSOR 15 (temperature double, humidity double)
  LOCALIZATION [47° 25’, 010° 59’]

- Registering a logical sensor network
  
  CREATE STREAM net_stream (id int, temperatur double)
  NETWORK [ 15 (6, 9)]
  SAMPLING 30 SECONDS
Web-based Box-and-Arrow Frontend

Example query:

- Detecting anomaly regions based on bursts

CREATE STREAM s_burst AS
SELECT timestamp, temperature
FROM net_stream [burst-detection(w => 1000, threshold =>'forecast')] ;

SELECT timestamp, temperature
FROM s_burst [anomaly-region (t => 0.5)]
Web-based Box-and-Arrow Frontend

- Example query:
  - Detecting anomaly regions based on bursts

AnduIN - A Declarative In-Network Data Mining Engine

DMSN - September 09

Developing and Deploying Sensor Network Applications with AnduIN
Query Decomposition and Optimization

**logical plan**

- sink
- anomaly region detection
- burst detection
- projection
- sampling
- source

**possible physical query plans**

- transformation
  - centralized
  - in-network

- possible physical query plans

DMSN - September 09

Developing and Deploying Sensor Network Applications with AnduIN
## Cost Model

- Costs for in-network query processing
- Factors influencing costs:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>total number of sensor nodes</td>
</tr>
<tr>
<td>$m_l$</td>
<td>number of leader nodes</td>
</tr>
<tr>
<td>$r_s$</td>
<td>sampling rate in $\frac{1}{s}$</td>
</tr>
<tr>
<td>$c_{wake}$</td>
<td>wake up costs</td>
</tr>
<tr>
<td>$c_{sample}$</td>
<td>sampling costs</td>
</tr>
<tr>
<td>$c_{msg}$, $\hat{c}_{msg}$</td>
<td>message costs</td>
</tr>
<tr>
<td>$h$</td>
<td>average tree height</td>
</tr>
<tr>
<td>$c_{cpu}$</td>
<td>local computation costs at a sensor</td>
</tr>
<tr>
<td>$c_{loc}$</td>
<td>costs for processing data from a neighborhood</td>
</tr>
<tr>
<td>$\sigma^j$</td>
<td>selectivity of operator $j$, $\sigma_0 = 1$</td>
</tr>
</tbody>
</table>
Cost Model

- Costs for in-network query processing
- No leader nodes

\[ C_{simple} = \begin{bmatrix} c_{wake} \end{bmatrix} \]
Cost Model

- Costs for in-network query processing
- Considering leader nodes

\[ C_{\text{innet}} = m \cdot ( \quad ) \]
Case Study

- Detect anomaly regions based on bursts
- 100 nodes
Summary

- AnduIN
  - Declaritive query formulation
  - Comprehensive optimization of complex queries
  - Cost model

- Future Work
  - Improving the implementation
  - Multi-query optimization
  - Online query optimization
Thank you...
## Case Study

- **Tmote Sky sensor nodes with**
  - 16 bit MCU MSP430F1611,
  - 4 MHz clock rate,
  - IEEE 802.15.4 compatible CC2420 transceiver with 250kBit/s

<table>
<thead>
<tr>
<th>Measurement</th>
<th>Time</th>
<th>Energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute average of 10 values</td>
<td>52.3μs</td>
<td>0.272μJ</td>
</tr>
<tr>
<td>Compute average of 100 values</td>
<td>245μs</td>
<td>1.274μJ</td>
</tr>
<tr>
<td>Single addition</td>
<td>2μs</td>
<td>0.010μJ</td>
</tr>
<tr>
<td>Single division</td>
<td>27μs</td>
<td>0.140μJ</td>
</tr>
<tr>
<td>Single multiplication</td>
<td>16.2μs</td>
<td>0.08μJ</td>
</tr>
<tr>
<td>Sending 1 byte</td>
<td>4.85ms(2.33 – 6.95ms)</td>
<td>240.19μJ(121 – 361μJ)</td>
</tr>
<tr>
<td>Sending 10 bytes</td>
<td>4.9ms(2.8 – 7.4ms)</td>
<td>252.93μJ(146 – 385μJ)</td>
</tr>
</tbody>
</table>