

Developing and Deploying Sensor Network Applications with AnduIN

Daniel Klan

Katja Hose

Kai-Uwe Sattler

Databases & Information Systems Group

Ilmenau University of Technology, Germany

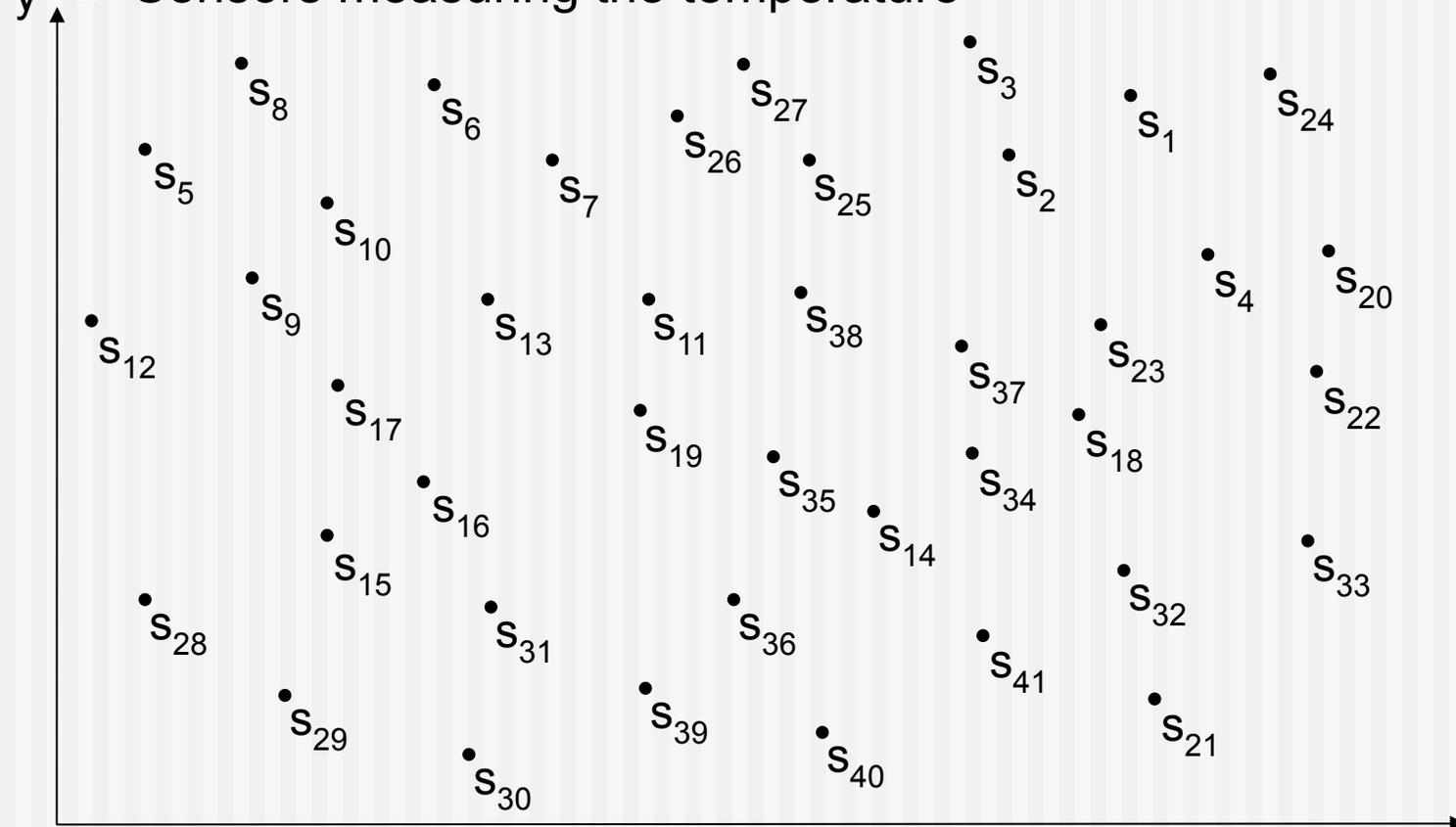
Outline

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

Introduction

- Anomalies in sensor networks

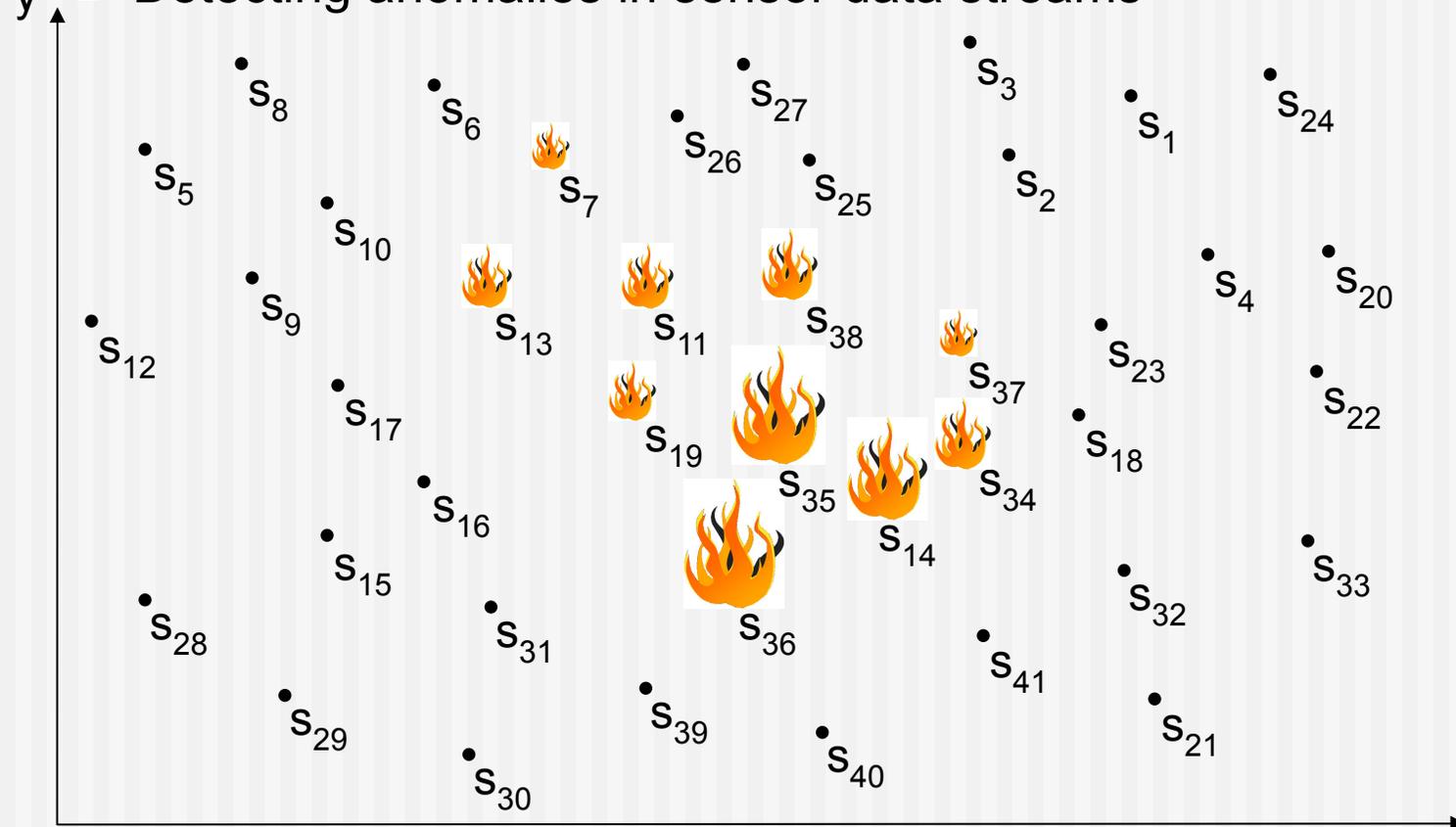
y ■ Sensors measuring the temperature



Introduction

- Anomalies in sensor networks

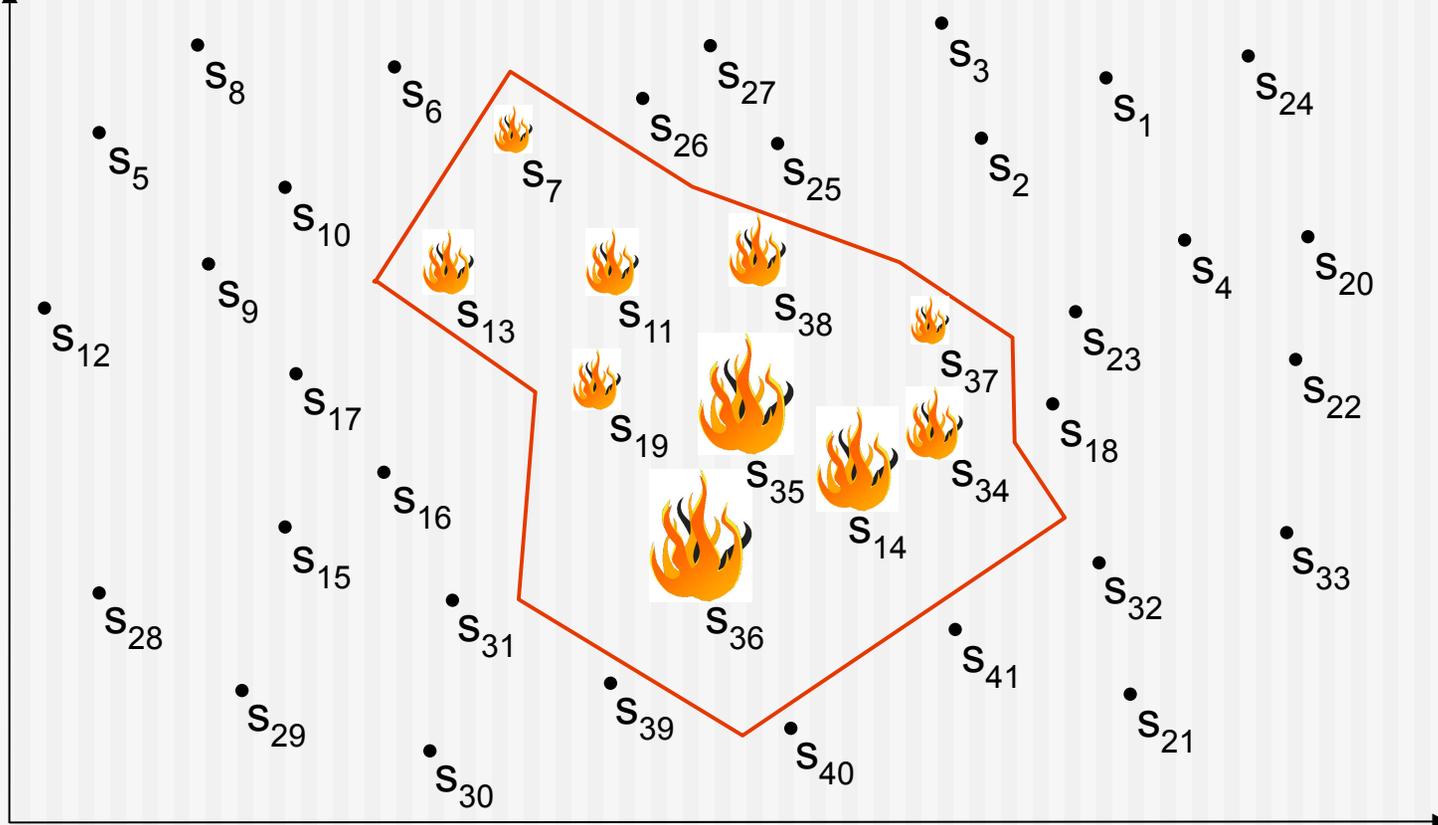
- Detecting anomalies in sensor data streams



Introduction

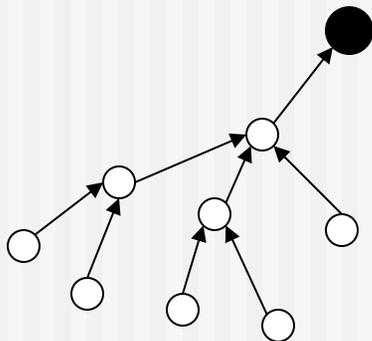
- Anomalies in sensor networks

y ■ Determine anomaly regions

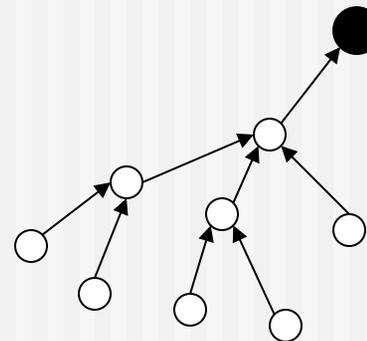


Introduction

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



Centralized processing

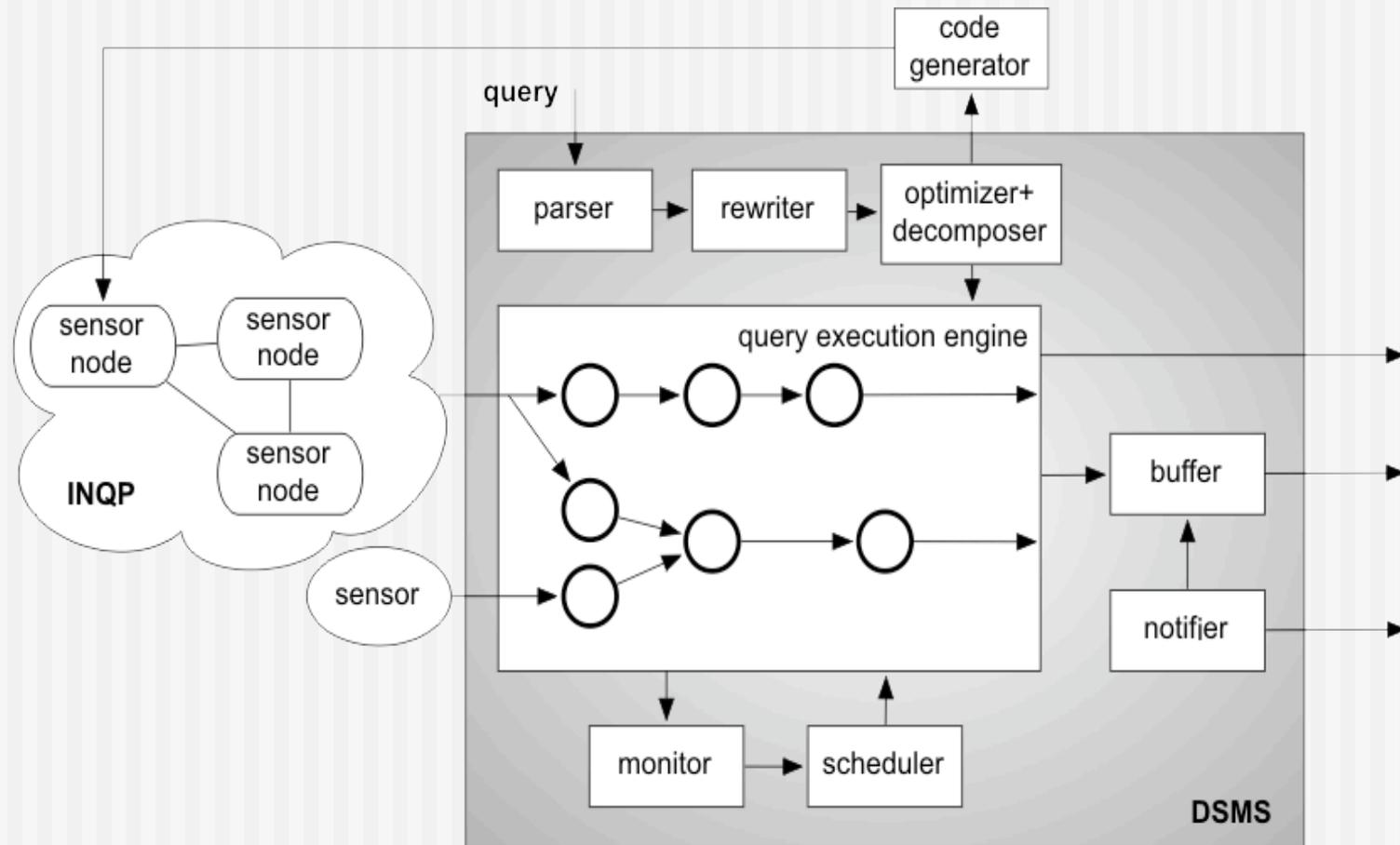


INQP

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

AnduIN



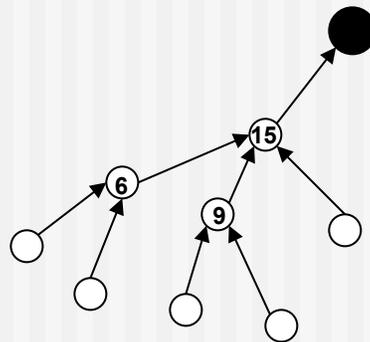
Logical Sensor Networks

- 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

Andu IN - A Declarative In-Network Data Mining Engine

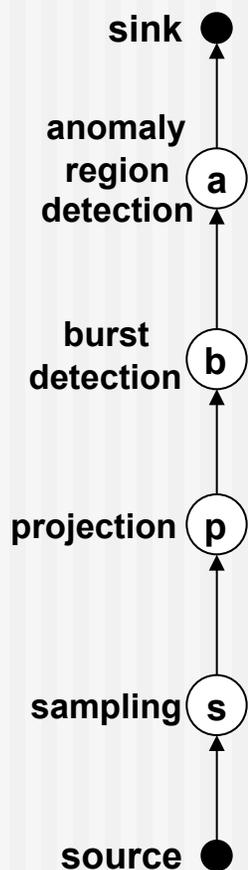
The screenshot displays the Andu IN web-based query editor interface. The main workspace shows a declarative query flow with the following components:

- netstream**: A data source node with parameters: id (int), v1 (double), v2 (double).
- Projection**: A transformation node with parameters: id (int), v1 (double, checked), v2 (double).
- Burst-Detection**: A data mining node with the following configuration:
 - Atributauswahl: v1 (checked)
 - Aggregation Tree Parameters: expanded
 - Forecasting: Methode (single smoothing), Steps (10), Factor (2)
- Anomaly-Region**: A data mining node with the following configuration:
 - Atributauswahl: v (checked)
 - Parameters: Threshold (0.7)
- Default**: A sink node.

The query flow is connected as follows: netstream feeds into Projection, which feeds into Burst-Detection, which feeds into Anomaly-Region, which finally feeds into the Default sink. A red oval highlights the 'Anomaly-Region' node and its connection to the 'Default' sink. Another red oval highlights the 'Burst-Detection' node and its connection to the 'Anomaly-Region' node. A third red oval highlights the 'netstream' and 'Projection' nodes. A fourth red oval highlights the 'netstream' node and the 'Anomaly-Region' node. The left sidebar shows a library of operators, with 'Anomaly-Region' and 'Burst-Detection' highlighted in red.

Query Decomposition and Optimization

logical plan



transformation



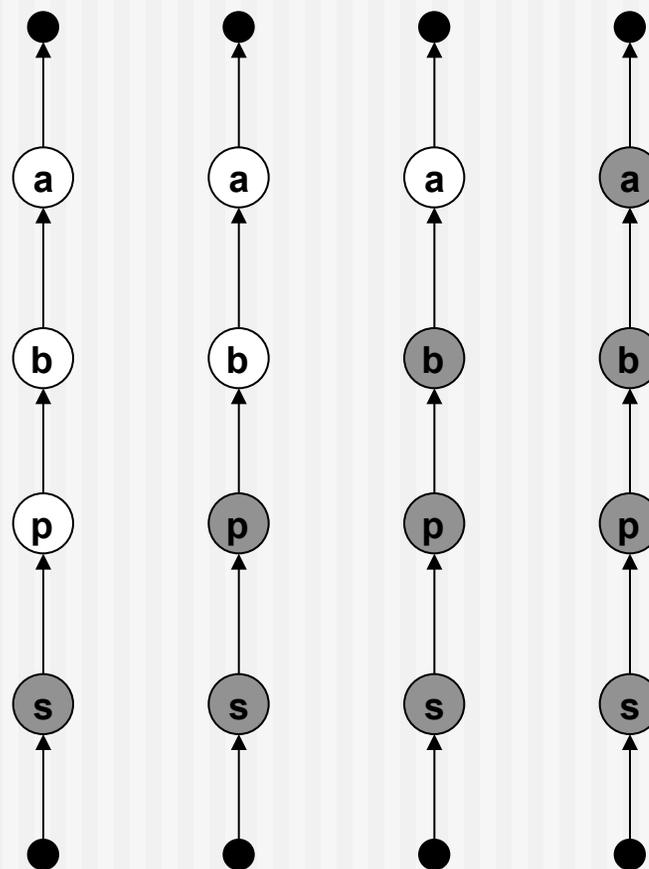
centralized



in-network



possible physical query plans



Cost Model

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

| | |
|--------------------------|---|
| m | total number of sensor nodes |
| m_l | number of leader nodes |
| r_s | sampling rate in $\frac{1}{s}$ |
| C_{wake} | wake up costs |
| C_{sample} | sampling costs |
| C_{msg}, \hat{C}_{msg} | message costs |
| h | average tree height |
| C_{loc}^{cpu} | local computation costs at a sensor |
| C_{neigh}^{cpu} | costs for processing data from a neighborhood |
| σ^j | selectivity of operator j , $\sigma_0 = 1$ |

Cost Model

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

$$C_{simple} = \left[c_{wake} \right]$$

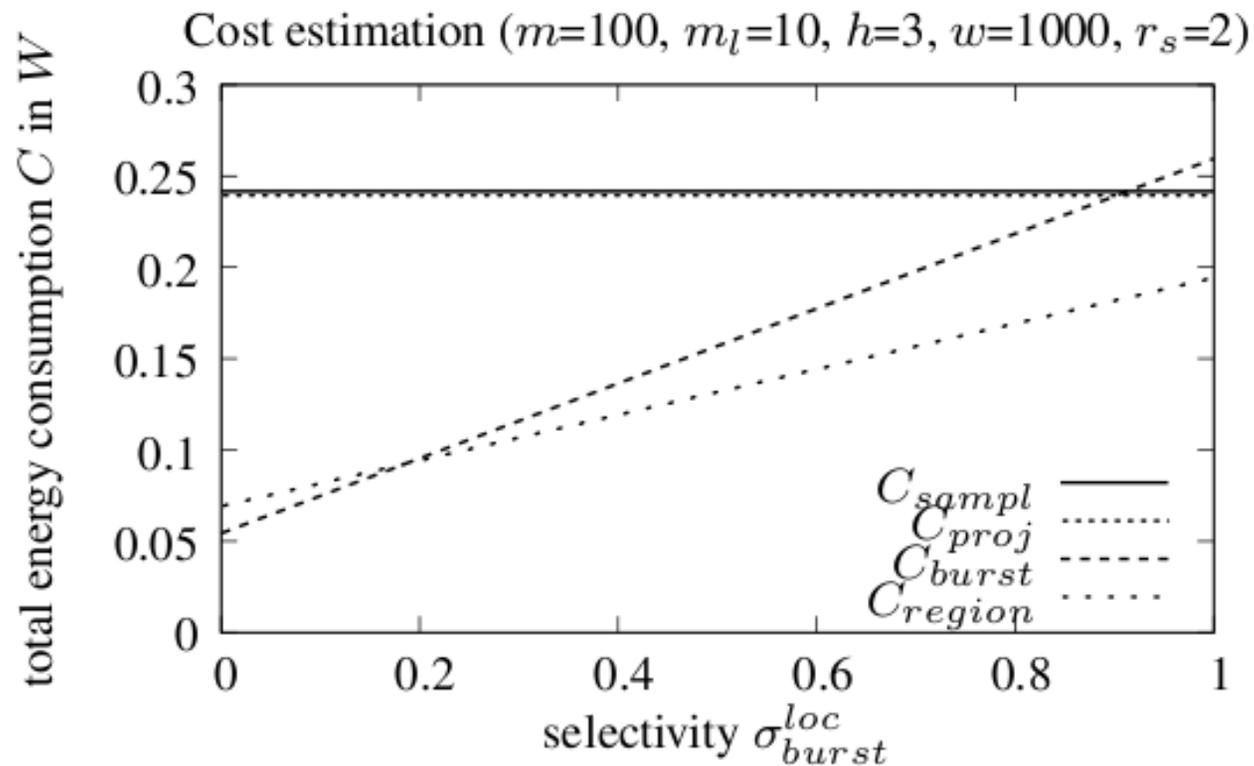
Cost Model

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

$$C_{innet} = \left[m \cdot (\right)$$

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

| measurement | time | energy |
|-------------------------------|-------------------------|-------------------------------|
| compute average of 10 values | $52.3\mu s$ | $0.272\mu J$ |
| compute average of 100 values | $245\mu s$ | $1.274\mu J$ |
| single addition | $2\mu s$ | $0.010\mu J$ |
| single division | $27\mu s$ | $0.140\mu J$ |
| single multiplication | $16.2\mu s$ | $0.08\mu J$ |
| sending 1 byte | $4.85ms(2.33 - 6.95ms)$ | $240.19\mu J(121 - 361\mu J)$ |
| sending 10 bytes | $4.9ms(2.8 - 7.4ms)$ | $252.93\mu J(146 - 385\mu J)$ |