Short Paper:
Location-Aware Event-Driven Query Processing
in Sensor Database Management

Soon J. Hyun
Prof. Department of Computer Science
Korea Advanced Institute of Science and Technology
Contents

• Introduction
• Related Works
• Event-driven query in SNQL
• Processing of Event-driven SNQL
• Evaluation
• Conclusion & Future Work
Sensor Database: WSN as database

Applications: habitat monitoring, disaster surveillance, military supports, patient monitoring, etc.
Introduction

- **Sensor Database**
  - Data from distributed sensor nodes
    - Spatial condition
  - **Continuous data**
    - Temporal condition
  - **Event-driven Query**
    - The type of queries which is executed by a sensor node *when specific event in the query is triggered*, not when it is arrived at the sensor node
    - Applications don’t have to monitor the dynamic occurrence of the events in the sensor network.
Related Works

• Researches on Sensor Database
  – Cougar
    • 2002, Cornell Univ.
  – TinyDB
    • 2006, UC Berkeley
    • Event-based / Periodic query
  – SNQL
    • 2008, iDB Lab., KAIST
    • Node selection / Conditional branching query
  – ELDQ
    • 2009, Wuhan Univ.
    • Event-driven query
Related Works

- **SNQL$^9$ Overview**

```sql
ON EVENT ( <event-condition> | <event-name> )
SELECT <select-list>
FROM <from-list>
WHERE <predicate>
SAMPLE PERIOD <period> FOR <duration>
WITHIN <participation-percentage>
CASE
  WHEN ( <event-predicate> | <event-name> ) THEN
    ( <query-stmt> | <select-clause> | <where-clause> | <sample-period-clause> )
  WHEN ... THEN
  ...
END
```
Contribution

- Limitations of event query in SNQL\textsuperscript{[9]}
  - No event definition query
  - No expression for spatial selective query dissemination
  - No expression for inner-query propagation

- Contribution
  - Improving event-driven query of SNQL\textsuperscript{[9]}
    - Temporal event monitoring condition (ON EVENT)
    - Spatial Selective dissemination (AROUND)
    - In-network propagation of inner-query (FROM)
  - Quadtree-based Distributed MBR Management
    - Maintaining Minimum Bounding Rectangle of each sub-tree
      - for selective query dissemination and in-network propagation
Event-driven Query in SNQL

- Event-driven Query in SNQL

\[
<\text{event-query}> ::= 
\]

\[
\text{ON EVENT} \ <\text{event-monitoring-clause}> \\
\text{AROUND} \ <\text{spatial-query-dissemination-condition}> \\
\text{SELECT} \ <\text{select-attribute-list}> \\
\text{FROM} \ <\text{inner-query-propagation-condition}> \\
\text{WHERE} \ <\text{predicate}> \\
\text{SAMPLE PERIOD} \ <\text{period}> \ \text{FOR} \ <\text{duration}> \\
\text{WITHIN} \ <\text{percentage}> \\
\text{CASE WHEN} \ <\text{conditional-branching}>
\]

\[
<\text{event-definition-query}> ::= 
\]

\[
\text{CREATE EVENT} \ <\text{event-name}> \ \text{WHEN} \ <\text{event-condition}>
\]
Event-driven Query in SNQL

• Query example

"An application wants to monitor a forest fire by analyzing the ambient data that are collected from the sensor nodes deployed over the mountain. When one of the sensor nodes on top of the mountain detects a particular situation about fire-alert from humidity readings, we want to collect humidity and wind data from some sensor nodes in the vicinity in order to closely watch the possibility of a fire.’ Let us suppose that the warning condition is when humidity is lower than 20%, and the coordination of mountaintop is (80, 40).

ON EVENT ( humidity ≤ 40% AND wind ≥ 5m/s, 5min, 30min )
AROUND BOUNDARY( (80, 40), 50m )
SELECT nodeID, humidity, wind
FROM BOUNDARY( Event.location, 10m )
Distributed MBR Management Methods

- **R-tree-based**
  - ELDQ uses the method for in-network propagation
  - Each of intermediate nodes holds MBRs of sub-trees
  - Overheads on R-tree by intersection region
    - during query dissemination & in-network propagation task

- **Quadtree-based**
  - Entire region (cell) is divided into four sub-cells recursively
  - Each cell has the root node which holds MBRs of sub-cells
  - It resolved the overheads in the R-tree-based method
Evaluation

• Simulation Settings
  – 500x500m square
  – 400 sensor node (random way-point)
  – (6 center points) x (3 radiuses) = 18 cases
  – 20 trials for each cases (average results)
  – 10% of disseminated sensor nodes become event nodes
    • It is fixed in each trial for three sets of queries
  – Input event-driven query in SNQL
    • ON EVENT (humidity ≤ 40 AND wind ≥ 5, 5min, 30min)
      AROUND BOUNDARY (<case1>~ <case18>)
      SELECT nodeID, humidity, wind
      FROM BOUNDARY( Event.location, 10m )
  – Corresponding queries of TinyDB / ELDQ are prepared
Evaluation

• **Simulation 1-1**
  
  – Total Transmission Count of event-driven queries in SNQL vs. TinyDB vs. ELDQ

  ![Graphs showing total transmission count in 30m, 50m, and 70m radius](chart)

  - SNQL has the least consumption energy for processing the query
Evaluation

• Simulation 1-2
  – Max Hop Count of event-driven queries in SNQL vs. TinyDB vs. ELDQ

➢ SNQL has the least time delay for processing the query
**Simulation 2**

- Performance of a single event-driven SNQL in the Quadtree model, compared with R-tree model

- QDMM has less *total transmission cost*, lower *max hop count* than R-tree.
- Intersection overhead of dissemination and propagation is saved
Conclusion & Future Work

• Improving event-driven query of SNQL
  – Temporal event monitoring condition (ON EVENT)
  – Spatial Selective dissemination (AROUND)
  – In-network propagation of inner-query (FROM)

• Quadtree-based Distributed MBR Management
  – Maintaining Minimum Bounding Rectangle of each sub-tree
    • for selective query dissemination and in-network propagation

• Future Work
  – More spatial operations in dissemination/propagation
  – Fully-mobile sensor database model
  – World Best Software Project *
Thank You
Questions and Comments
Appendix-A

R-tree vs. Quadtree: Dissemination
Appendix-A

R-tree vs. Quadtree: Propagation

R-tree

Quadtree
Three queries in SNQL, TinyDB and ELDQ

An application wants to monitor a forest fire by analyzing the ambient data that are collected from the sensor nodes deployed over the mountain. When one of the sensor nodes on top of the mountain detects a particular situation about fire-alert from humidity readings, we want to collect humidity and wind data from some sensor nodes in the vicinity in order to closely watch the possibility of a fire.’ Let us suppose that the warning condition is when humidity is lower than 20%, and the coordination of mountaintop is (80, 40).

Equivalent queries in SNQL

```
ON EVENT ( humidity ≤ 40% AND wind ≥ 5m/s, 5min, 30min )
AROUND BOUNDARY( (80, 40), 50m )
    SELECT nodeID, humidity, wind
FROM BOUNDARY( Event.location, 10m )
```
## Appendix-B

### Equivalent queries in TinyDB

A. **SELECT** nodeID  
   **FROM** sensors  
   **WHERE** distance( location, (80, 40) ) < 50m

B. **ON EVENT** fire-detected  
   **SELECT** location, nodeID, humidity, wind  
   **FROM** sensors  
   **WHERE** nodeID = { result of (a) }

C. **SELECT** nodeID, humidity, wind  
   **FROM** sensors  
   **WHERE** distance( location, { location results of (a) } ) < 10m

**Condition:**  
humidity ≤ 40%  
wind ≥ 5m/s  
**Monitoring-Period/Duration:**  
5 min / 30 min
Appendix-B

• Equivalent queries in ELDQ

A. SELECT s.nodeID
   FROM sensors AS s
   INSIDE ( 50m, (80, 40) )

B. SELECT s.nodeID, s.humidity, s.wind
   FROM sensors AS s
   INSIDE ( 10m, ( SELECT e.location FROM sensors AS e
       WHERE e.nodeID = { result of (a) } ) )
   START ON ( e.humidity ≤ 40% AND e.wind ≥ 5m/s )
   SAMPLE PERIOD 5 min FOR 30 min