효율적인 센서 네트워크 관리를 위한 다중 연속질의 분할

박정업*, 조명현*, 손진현*
*한양대학교 컴퓨터공학과
e-mail : {jupark, mhjo, jhson}@cse.hanyang.ac.kr

The Multiple Continuous Query Fragmentation for the Efficient Sensor Network Management

Jung-Up Park*, Myung-Hyun Jo*, Jin-Hyun Son*
*Dept. of Computer Engineering, Hanyang University

요 약

최근 센서네트워크에 관련된 많은 연구가 진행되고 있다. 특히, 센서의 전력 보전을 위한 많은 기술들이 개발되고 있는데, 본 논문에서는 센서 네트워크의 불필요한 전력 소비를 줄이는 다중 연속질의 최적화에 관련된 방법을 제시한다.

우리는 센서 네트워크에서 전송되는 데이터의 횟수나 전송량의 원천적 문제가 되는 다중 연속 질의의 중복성 문제를 해결하는 분할 알고리즘을 제안한다. 분할 알고리즘은 새롭게 생성된 사용자 질의와 기존의 질의 들 사이에 질의 중첩 질의 영역을 제거하기 위해, QR-트리 기반의 질의 인덱스를 통해 하나의 질의를 둘 이상의 질의로 분할하는 알고리즘이다. QR-트리는 효율적인 질의 분할을 위해, R*-트리를 본 논문의 구조에 맞게 개량한 것이다.

1. 서론

최근 몇 년간, 네트워크와 모바일 컴퓨팅에 관한 관심이 상당한 연구 분야로 다가오고 있다. 특히 특정 지역의 관찰을 위해 많은 센서들로부터 애드-혹(ad-hoc) 데이터를 추출하는 무선 센서 네트워크에 관심이 집중되고 있다.

무선 센서 네트워크에 관련된 연구는 크게 두 가지 분야로 나뉘어 진행 중이다. 첫째, 센서 및 센서 네트워크를 구축, 관리하는 연구 분야이다. 이것은 적은 에너지 소모를 위한 저전력(low-power) 센서를 개발할 뿐만 아니라, 센서 네트워크에 적용되는 토폴로지 및 프로토콜도 개발한다. 특히 센서는 제한된 배터리를 갖기 때문에, 센서 수명 연장에 관련된 연구가 이 분야에서 가장 주목할 만한 논제라고 할 수 있다. 둘째, 센서로부터 전송된 데이터를 처리하고, 센서로 전송될 연속질의를 처리하기 위한 서버 측면의 연구가 진행되고 있다.

본 논문은 센서 네트워크 두 분야의 중간에 위치한 연구로써, 본 연구의 궁극적인 목적은 센서의 수명을 연장시키기 위한 것이다. 하지만 기존 연구처럼 센서 네트워크에 구성된 센서들 사이의 토폴로지나 프로토콜에 관련된 알고리즘을 개발하는 것이 아니라, 사용자에서 센서로 전해지는 다중 연속질의의 중복 영역을 최적화하여 센서의 불필요한 전력 소모량을 최소화하고자 한다. 다중 연속질의들 사이의 중첩 문제를 해결하기 위해, 본 논문에서는 질의 인덱스 QR-트리를 제안한다.

본 논문의 구성은 다음과 같다. 제 2장에서는 기존 연속질의 최적화 방법을 기술한다. 제 3장에서는 기반 기술인 센서 네트워크와 연속질의에 대해 간단히 기술하고, 본 연구의 동기를 기술한다. 제 4장에서는 다중 연속질의 분할의 세 단계를 기술하고, 제 5장에서 결론을 맺는다.

2. 관련 연구

연속질의는 센서 네트워크가 출현하기 이전부터, 다양하게 연구되었다. Terry[3]는 첨부(append) 기반의 관계형 데이터베이스에서 연속질의를 처리하는 연구를 제안했고 OpenCQ[4]는 분산 환경의 정보를 수집하기 위해 연속질의를 사용하였다. 하지만, Tapestry와 OpenCQ에는 모두 연속질의에 대한 최적화가 논의되지 않았다.

Sellis는 다중 연속질의에 대한 최적화를 위해, 질의의 각 조건식(predicate)들의 관계성을 정립하여 총체적 계획(global plan)을 구성하였다. 하지만 총체적 계획 때문에, 각 질의 predicate가 서로 반대로 내포할 수 있는 경우를 처리할 수 없고, 내포가 아닌 일부 중복에 대한 최적화도 할 수 없다. Chen이 제안한 NiagaraCQ[1]는 Sellis가 제안한 질의의 최적화를 동적인 상황에 맞게 최적화하기 위한 단계적 증식 방법을 제안했다. Madden의 CACQ[2]는 NiagaraCQ보다 좀 더 유연한 방법으로 다중 연속질의를 그룹 핑하였다. 마지막으로, Yousuke[5]가 제안한 다중 연속질의 최적화 방법이 본 논문의 다중 연속질의 최적화 방법과 유사한데, Yousuke는 다중 연속질의의 시간 범위를 유사성 행렬을 이용해 클러스터링 시킨 후, DAG에 의

해 질의 계획(Query Plan)을 최적화 시켰다.

센서는 제한된 컴퓨팅 능력과 파워를 갖기 때문에, 다중 연속질의의 최적화 시스템을 모두 장착한다는 것은 현실적으로 불가능하다. 그러므로 센서의 저전력을 위해서는 질의가 센서에 전달되기 이전에, 다중 연속질의의 최적화를 위해 질의의 재구성이 수행되어야 한다.

3. 기반 기술 및 문제 정의

본 장에서는 본 연구에서 가정하는 무선 센서 네트워크와 센서 네트워크에서 사용하는 TinyDB의 연속질의(Continuous Query)에 대해 간단히 기술한다.

3.1 무선 센서 네트워크 (Wireless Sensor Networks)

무선 센서 네트워크(WSN)는 많은 수의 작고 이질적인 센서 노드들 간의 네트워크를 의미하는데, 각 센서 노드들은 전원 유닛, 센싱 유닛, 컴퓨팅 유닛, 그리고 통신 유닛을 포함하고 있다.

지금까지 많은 센서들이 개발되었는데, 본 연구에서는 Greate Duck Island 와 James Reserve[6]에 설치되어 연구된 적이 있는 Mica 플랫폼[6]을 기반으로 한다. 센서는 제한된 배터리를 갖는데, 센서에서 다음의 세 가지 부분이 가장 많은 에너지를 소모한다[6,7]. 이웃 노드에 패킷을 전송하는 요소(sending cost is 1μJ/bit [7]), 패킷을 전송 받는 요소(sending cost is 0.5μJ/bit [7]), 전송받는 패킷을 버퍼에 쓰는 요소(writing/erasing cost is 1μJ/byte [7])이다.

3.2 연속질의 (Continuous Query in TinyDB)

TinyDB[8]는 TinyOS[9]를 기반 한 센서들로부터 정보를 추출하기 위한 질의 처리 시스템이다. TinDB는 간단한 SQL 질의 인터페이스를 제공하며 연속적인 데이터를 추출할 수 있는 연속질의(Continuous Query) 형태로 구성된다. 연속질의는 그들의 실행에 사용되는 범주에 따라 두 가지로 분류된다. 하나는 측정 데이터의 변경에 따른 변동(Change)-기반 연속질의이며, 다른 하나는 일정 주기마다 질의를 실행시키는 시간(Timer)-기반 연속질의이다[1]. 본 논문의 센서 네트워크는 자연 현상을 계속적으로 관찰하기 위한 것이기 때문에, 시간 기반 연속질의를 사용한다. 그림 1은 사용자가 관찰 시간을 지정할 수 있도록, TinyDB에 사용되는 연속질의[8]이다. 본 논문은 TinyDB 연속질의의 전체 스키마를 이용하지 않고, 그림 1과 같은 제한된 질의에 대해서만 질의 최적화를 제공한다.

```
SELECT *
FROM Sensors
WHERE {A op constant}*
[SAMPLE INTERVAL period]
[LIFETIME period]
```

```
SELECT *
FROM Sensors
WHERE 온도 < 30 and 습도 > 20
SAMPLE INTERVAL 30 seconds
or
LIFETIME 2 days
```

(그림 1) 시간 기반 연속질의

TinyDB는 'Sensors'라는 하나의 테이블 만으로 구성되며, 센서 노드가 포함하는 센서 타입들은 'Sensor' 테이블의 컬럼에 해당한다. TinyDB의 이런 특징들 덕분에 본 논문도 복잡한 질의에 대한 최적화는 고려하지 않는다. 센서로부터 측정된 데이터는 'Sensors' 테이블에 튜플로 주기적으로 저장된다. SAMPLE INTERVAL 절과 LIFETIME 절은 질의 결과를 서버에 연속적으로 반환하기 위한 요소로써, 두 절중 하나만이 선택적으로 사용될 수 있다.

물리적 환경의 데이터를 측정하는 센서 네트워크에서 센서들은 제한된 도메인을 갖는다. 예를 들어, 만일 온도 센서가 영하 50℃나 영상 100℃ 이상의 값을 측정한다면, 센서 장치는 작동하지 않을 것이다. 제한된 도메인은 WHERE 절의 각 조건식(Predicate)을 가정 1로써 새롭게 정의할 수 있다. 이것은 질의 영역을 hyper-rectangle 형태로 구성하기 때문에, 다중 연속질의들 사이에 비교를 쉽게 해준다.

가정 1. 바운드 조건식 (t)

레코드가 $R[A_1, A_2, .., A_n]$ 라면,
바운드 조건식(t_j)은
t_j : lValue Θ A_i Θ rValue ,
where Θ ∈ { <, >, ≤, ≥},
lValue, rValue ∈ D_i ,
D_i는 A_i의 도메인이고, lValue ≤ rValue

WHERE 절에 정의되지 않는 필드를 가정 1에 의해, 시스템 하한 값과 상한 값을 정의한다면 두 질의는 모두 동일한 필드를 갖고 질의하기 때문에 비교하기 쉽다. 가정 1에 의해 습도와 온도가 잠재적으로 상한 값(100)과 하한 값(0)을 갖는다고 가정하면, 그림 2는 그림 3처럼 재구성(rewriting)될 수 있다. 그림 3의 Q1′와 Q2′의 WHERE 절은 모두 공통의 바운드된 필드로 구성되었기 때문에, 비교하기 쉽고 두 질의의 중복성을 간단하게 판단할 수 있다.

Q1	Q2
SELECT * **FROM** Sensors **WHERE** *temperature* > 30 and *humidity* > 20 **SIMPLE PERIOD** 1024	**SELECT** * **FROM** Sensors **WHERE** 50 > *temperature* **SIMPLE PERIOD** 1024

(그림 2) 일반적인 연속질의의 예제

Q1′	Q2′
SELECT * **FROM** Sensors **WHERE** 30<*temperature*<100 and 20<*humidity*<100 **SIMPLE PERIOD** 1024	**SELECT** * **FROM** Sensors **WHERE** 0<*temperature*<50 and 0<***humidity***<100 **SIMPLE PERIOD** 1024

(그림 3) 재구성된 연속질의의 예제

3.3 문제 정의 (Problem Formulation)

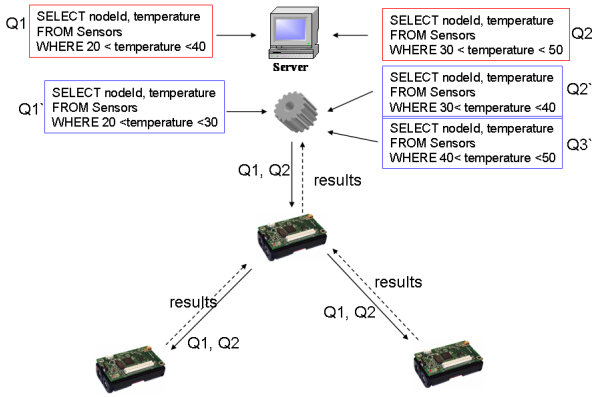
본 절은 본 논문에서 가정한 센서 네트워크에서 연속질의를 처리함으로써 발생할 수 있는 문제점을 정형화한다. 물리적 환경을 측정하기 위해서는 모든 센서들이 동일한 질의를 처리해야 한다. 처리된 질의 결과는 패킷 형태로 서버에 전달되는데, 계산의 용이성을 위해 하나의 튜플이 한 개의 패킷으로 변환되어 전송된다고 가정한다. 튜플은 그림 4와 같은 형태처럼 패킷화 되어있다[8,9].

SELECT *node,light*
FROM *sensors*

Field Name : Size (bytes)	QID : 1	numFields : 1	notnull : 4	field 1 : size(1)	...	field n : size (n)
Sample Data For Query	1	2	110...0	N1 N2 L1 L2		

(그림 4) TinyDB에 저장된 튜플 포맷[8]

센서 네트워크에서 중복된 영역의 질의들은 센서에게 중복된 패킷을 전송하게 함으로써 불필요한 에너지를 소모하게 한다. 사실 센서 네트워크는 일반적으로 그림 5와 같은 트리 구조[8,9]로 구성되기 때문에, 중복된 패킷이 발생한다면 발생한 센서 노드뿐만 아니라, 노드의 상위 노드들도 불필요한 에너지를 소모하게 된다. 즉, 센서 네트워크의 생명 주기를 짧게 하는 문제를 초래한다



(그림 5) 센서 네트워크의 질의 처리

질의 영역은 센서에서 제공하는 필드의 개수에 따라 n 차원의 유클리디언 공간으로 표현될 수 있다. 그래서 질의 영역 R은 (S,T)인 두 꼭지점으로 표현될 수 있는데, S=[s1 ,s2 , ... , sn] 이고 T=[t1 , t2 , ... , tn], si ≤ ti for 1 ≤ i ≤ n 이다. 이러한 정의를 기반으로 두 질의 영역이 중첩되어 있다는 것은 정의 1처럼 명시할 수 있다.

정의 1. 중첩성 (Ω)

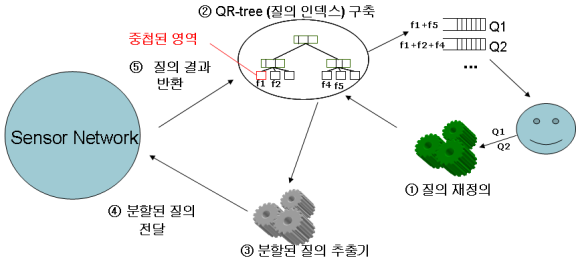
두 질의 영역 R_i 과 R_j 가 중첩되어 있다는 것은 아래의 조건을 만족한다는 것을 의미한다.

$$\forall D_i = true \ and \ D_i = \left\{ | C_1^i - C_2^i | < \frac{L_1^i + L_2^i}{2} \right\}$$

1≤ i ≤ n, n 은 필드의 개수이며,
L = t_i − s_i ,C = (t_i + s_i)/2

4. 다중 연속질의 분할 (Multiple Continuous Query Fragmentation)

다중 연속질의들 사이의 중복성을 제거하기 위해서는 몇 가지 단계가 필요하다. 일단 질의들 사이의 중복된 영역을 쉽게 비교하기 위해서 질의들을 동일한 형식으로 재구성(Wrapping) 할 필요가 있다. 재구성 단계 이후에서는 질의 인덱스에 질의를 삽입하는 단계가 필요하다. 그 이후에 질의 인덱스에 질의를 삽입하여 질의 인덱스를 재구성하게 되면, 실제 센서에 전달할 분할 질의를 추출하는 단계가 필요하다. 이런 과정으로 센서 네트워크에 전달된 질의들은 질의에 따라 결과를 베이스스테이션에 전달하게 된다. 전달된 질의 결과는 원본 결과로 사용자에게 반환되기 위해 질의 인덱스를 통해 필터링 및 병합이 이루어진다.



(그림 6) 다중 연속질의 분할 전략

중첩된 질의들을 분할할 때 고려되어야 하는 요소는 크게 두 가지이다. 하나는 분할된 질의의 개수를 최소화하는 것이다. 그 이유는 분할된 질의 개수가 많아질수록, 센서에게 불필요한 프로세싱이 요구되어 부가적인 에너지 소모를 유발할 수 있기 때문이다. 원본 질의의 개수와 변경이 없으면서, 중첩된 질의 영역을 제거하기 위한 방법이 연구되어야한다. 다른 하나는, 연속질의는 오랜 시간동안 질의에 대한 결과를 반환하기 때문에, 베이스스테이션에 상당히 많은 연속질의들이 존재하게 될 것이다. 그래서 삽입 질의와 베이스스테이션에 존재하는 모든 질의들과 중첩된 영역을 비교하는 것은 상당히 많은 시간을 필요로 하기 때문에, 최대한 빨리 분할 질의를 추출할 수 있는 연구가 필요하다. 본 연구는 두 요소 중 두 번째 요소를 해결하기 위한 질의 인덱스(QR-tree)를 제안한다.

4.1 질의 재구성

다중 질의를 비교하기 쉽게 구성하는 재구성 단계는 크게 세 단계로 구성된다. 첫 번째 단계로, 가정 1에 의해 다중 질의들 사이의 WHERE 절에 모두 동일한 필드를 갖도록 시스템 상한 값과 하한 값을 결정하는 것이다. 두 번째 단계로, WHERE 절의 OR 연산자를 제거하기 위해 WHERE절의 조건식을 DNF (Disjunctive Normal Form) 형태로 변경해야 한다. OR 연산자로 구성된 질의에 대한 결과는 OR 연산자마다 분할된 질의 결과의 합과 동일하기 때문에, DNF로 구성된 조건식을 OR 연산자에 따라 질의를 분리해 OR 연산자를 제거해야 한다. 마지막 단계로, 연속질의의 시간 절(LIFETIME과 SAMPLE PERIOD)을 분리할 필요가 있다. 즉, LIFETIME 절로 구성된 질의들과 SAMPLE PERIOD 절로 구성된 질의들로 구별하여 다중 질의에 대한 인덱스를 구축할 필요가 있다. SAMPLE PERIOD는 지정된 값의 최대 공약수 형태로 질의들을 클러스터링 해야 한다.

4.2 QR-트리

QR-트리는 질의들 사이의 중첩 영역을 빨리 제거하기 위해 R*-트리[10]를 응용한 질의 인덱스이다. QR-tree는 R*-트리[10]가 가지고 있는 일반적인 속성을 가지고 있다. 정의 2는 버킷과 노드가 포함하는 속성을 의미한다.

정의 2. 버킷(B) 과 노드(n)

B = { N, B, m }
where N: 노드(n)들의 집합, B: 링크로 연결된 버킷
n = { R, Q, M, B }
where R: 노드의 질의 영역, Q: 원본질의 집합,
B: 자식 버킷 M: 변경자

버킷은 최대 m 개수만큼 노드들을 포함하는데, 리프 버

킷의 경우 이웃 버킷의 레퍼런스를 포함한다. 노드는 질의 영역을 통해 질의를 대변하는데, 인덱스 노드들은 자식 버킷의 MBR을 자신의 질의 영역으로 갖는다. 인덱스 노드들은 자식 노드들의 영역을 모두 포함하기 때문에 질의 영역(R)에 따라 가지치기가 가능한 것이다. 원본질의 집합(Q)은 리프 노드들이 갖는 속성으로써, 사용자가 전달한 실제 질의를 참조한다. 이것은 분할 질의 생성 및 질의 결과 반환 시 이용된다. 변경자(M)는 리프 노드들만 갖는 속성으로써, 질의 인덱스에서 변경된 노드들만 센서 네트워크에 전달해야 하기 때문에, 노드가 변경되었는지 유무를 파악한다. 자식 버킷(B)은 인덱스 노드들만 갖는 속성으로써, 탐색을 용이하게 한다.

버킷의 MBR은 버킷에 포함된 노드들의 모든 영역(R)들을 포함하는 최대 바운더리를 의미하고, 이것은 연결된 상위 노드의 질의 영역(R)에 대변하게 된다. 질의 영역(R)은 탐색의 기준을 의미하기 때문에 최대한 인덱스 노드들끼리 중첩 영역이 없도록 구성해야 잘못된 곳을 찾아가는 일을 줄일 수 있다. 즉, 버킷에 m+1개의 노드수가 포함되어 분할하게 될 경우, 노드들을 적절하게 버킷에 분배해야 한다. 만일 버킷의 MBR을 불필요하게 크게 구성할 경우, 중첩된 질의 영역을 찾는데 오랜 시간이 걸려 결과적으로 분할 질의를 획득하는 데 오랜 시간을 소요하게 된다. 그래서 본 연구에서는 R*-트리의 장점을 받아들여 최대한 버킷의 MBR에서 여백(margin)을 최소화시킬 수 있도록 버킷을 구성하고자 한다. 만일 여백이 동일할 경우에는 두 집합의 넓이 합이 최소가 되도록 구성한다. 여백을 적게 한 이유는 탐색 실패율(fault rate)을 줄이기 위함이고, 넓이를 최소화하는 것은 영역들을 균등하게 분포시키게 위함이다. 리프 노드에서 R*-트리처럼 오버랩(overlap)의 요소는 고려하지 않고 여백과 넓이(area)만을 고려한 것은 QR-트리의 구조가 R*-트리의 구조와는 달리 리프 노드들의 영역이 모두 disjoint한 관계를 갖기 때문이다.

4.3 분할된 질의 추출기 (Fragmented Query Extractor)

QR-트리로 구성된 질의 인덱스에서 리프 노드들은 센서 네트워크에 전달할 질의들을 의미한다. 새롭게 삽입된 질의는 리프 노드와의 중첩 관계를 판단하여 새롭게 인덱스에 포함된다. 삽입된 질의는 리프 노드에 존재하게 되는데, 이 과정 중에 기존 리프 노드들을 변경하거나 삭제하여 새롭게 센서 네트워크에 전달해야 하는 질의들을 발생시킨다. 그래서 센서 네트워크에 새롭게 삽입될 질의들은 일단 리프 노드들의 변경자가 체크된 노드들만을 고려한다. 리프 노드들은 리프 버킷에 포함되며, 리프 버킷은 정의 2와 같이 이웃 버킷을 레퍼런스하고 있기 때문에, 리프 버킷들을 탐색하면서 센서 네트워크에 삽입되어야 할 리프 노드들을 추출할 수 있다. 질의 인덱스에서 분할된 리프 노드의 개수가 많기 때문에, 분할된 모든 리프 노드들을 질의로써 센서 네트워크에 전달하는 것은 많은 시간과 비용을 필요로 한다. 그래서 되도록 질의의 개수를 줄이기 위해, 질의 재구성 단계에서 OR 연산자에 사용한 방법을 역으로 이용한다. 원본질의 집합(Q)이 같은 리프 노드들은 동일한 질의들에게 결과를 반환하기 때문에, OR 연산자로 병합하여 하나의 질의로 구성할 수 있다. 질의들의 영역은 hyper-rectangle 형태이기 때문에, 분할되는 질의 영역들도 hyper-rectangle로 구성하기 위해 축을 기반으로 분할하였다. 즉, 중첩되지 않는 질의 영역들도 분할되어 많은 질의 분할 노드들을 생성시킨 것이다. 그래서 센서 네트워크에 전달할 질의들은 리프 노드들을 그대로 전달할 필요가 없다.

5. 결론 및 향후 과제

센서들은 중복된 데이터를 반환함으로써 불필요한 에너지를 소모하게 된다. 이러한 문제점을 해결하고자 중복된 데이터의 원천적인 문제가 되는 다중 연속질의들 사이의 중복성을 해결하고자 하였다.

다중 연속질의들 사이의 중복성은 WHERE절의 중첩된 조건식을 차원 축을 기반으로 분할하는 알고리즘을 통해 간단히 해결할 수 있었다. 하지만 차원이 증가하거나 베이스스테이션에 존재하는 질의가 증가함에 따라 분할에 필요한 처리 시간이나 분할된 질의 개수가 많아지는 문제를 초래하였다. 그래서 본 연구는 후자보다는 전자의 문제점이 먼저 해결되어야 한다고 판단되어, R*-트리 기반의 QR-트리를 제안하였다. QR-트리는 R*-트리의 정책을 받아들여, 버킷의 여백을 최소화하도록 구성하여 빠르게 중첩된 질의를 발견할 수 있도록 하였다.

하지만 본 연구에서 구성한 분할 알고리즘은 아직 분할 질의 개수가 많은 문제점을 갖고 있다. 현재까지의 연구는 질의에 대한 결과는 고려하지 않았다. 그래서 향후 반환된 데이터의 시놉시스를 기반으로 분할 질의 개수를 줄일 수 있는 방법을 연구하고자 한다.

7. 참고 문헌

[1] J. Chen, D. DeWitt, F. Tian, Y. Wang. NiagaraCQ: A Scalable Continuous Query System for Internet Databases. In Proc. ACM Int. Conf. on Management of Data, 2000, pp. 379-390.

[2] S. Madden, M. Shah, J. Hellerstein, V. Raman. Continuously Adaptive Continuous Queries Over Streams. In Proc. ACM Int. Conf. on Management of Data, 2002, pp. 49-60.

[3] D. Terry, D. Goldberg, D. Nichols, and B. Oki. Continuous queries over append-only databases. In Proc. of the 1992 ACM SIGMOD Intl. Conf. on Management of Data, pages 321-330, June 1992.

[4] L. Liu, C. Pu, and W. Tang. Continual queries for internet scale event-driven information delivery. IEEE Trans. on Knowledge and Data Engineering, 11(4):583-590, Aug. 1999.

[5] Yousuke Wantanabe. Hiroyuki Kitagawa. A Multiple Continuous Query Optimization Method Based on Query Execution Pattern Analysis. DASFAA 2004, LNCS 2973, Pages 443-456.

[6] A. Mainwaring, J. Polastre, R. Szewczyk, and D. Culler. Wireless sensor networks for habitat monitoring. In ACM Workshop on Sensor Networks and Applications, 2002.

[7] J. Hill, R. Szewczyk, A.Woo, S. Hollar, and D. C. K. Pister. System architecture directions for networked sensors. In ASPLOS, November 2000.

[8] Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. The Design of an Acquisitional Query Processor for Sensor Networks. To Appear, SIGMOD, June 2003.

[9] TinyOS : www.tinyos.net

[10] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The R*-tree: An efficient and robust access method for points and rectangles," Proceedings of ACM SIGMOD Int'l. Conf. on Management of Data, pp. 322-331, 1990.