SenDB: 무선 센서 네트워크용 질의 처리 시스템

김민규° 김도혁 김태형
한양대학교 컴퓨터공학과
{mgkim°, dohyuk, tkim}@cse.hanyang.ac.kr

SenDB: Query Processing System for Wireless Sensor Network

Minkyu Kim° Dohyuk Kim Tae-Hyung Kim
Department of Computer Science and Engineering, Hanyang University

요 약

자원 제약적인 무선 센서네트워크상에서 전송비용을 최대한 줄이기 위하여 데이터의 수집 및 처리를 분산된 형태로 처리하는 방법이 필수적이다. 또한 Declarative Query를 이용하여 다양한 질의를 표현하고 처리할 필요성이 있다. 센서네트워크 데이터베이스는 이와 밀접한 관련이 있고 본 논문에서는 기존의 센서네트워크 데이터베이스 시스템의 문제점을 분석해보고 해결책을 제안하고자 한다. 아울러 유한 상태 머신 기반의 실행모델을 이용하고 응용의 변화에 대처할 수 있게 동적 재구성 기능을 지원하도록 설계된 SenOS상에서 센서네트워크 질의 처리 시스템의 구조와 설계방법에 대하여 살펴보겠다.

1. 서 론

무선 센서 네트워크는 기존의 컴퓨팅 시스템과는 달리 극도로 제한된 시스템 자원과 열악한 환경 속에서 동작하며 저 전력 마이크로프로세서와 무선통신 모듈, 센서 등을 통합한 시스템이 내장된 센서 노드를 이용하여 구성하게 된다. 이러한 무선 센서 네트워크는 현실 세계의 다양한 환경에 대한 각종 정보를 실시간으로 수집하고, 처리함으로써 다양한 응용에 사용되어 질 수 있다. 특히 사람의 접근이 불가능한 취약지구나 지속적인 관리가 어려운 지역, 예를 들어 사회기반시설의 안전감시, 산불 감시, 산업시설 감시, 국방 등의 분야에서 무선 센서 네트워크가 효율적으로 사용되어 질 수 있다[1]. 하지만 무선 센서 네트워크가 이런 응용에 사용되기 위해서는 각 센서 노드에 내장되어 있는 센싱 장치들이 물리적 신호, 즉 열, 소리, 압력 등에 반응하여 데이터를 추출해야 할 뿐만 아니라 각각의 노드는 이웃 노드 혹은 베이스스테이션으로 센싱된 데이터를 전송할 수 있어야 한다. 따라서 무선 센서 네트워크상에서 물리적인 현상에 대한 정보들을 모으는 방법에 대한 연구가 확대되고 있다.

기존의 무선 센서 네트워크상에서 센싱된 데이터를 모으는 방법은 중앙 집중형 시스템에 주로 의존해 오고 있었다. 즉 모든 센서노드들이 개발자에 의해 미리 프로그래밍 되어 다양한 환경에 뿌려져야 했으며, 일단 뿌려진 노드들은 프로그래밍 된 동작 이외의 것은 수행할 수 없는 시스템이었다. 또한 무선 센서 네트워크상에서 환경에 대한 간단한 정보를 추출하기 위해서는 수백 혹은 수천 줄의 프로그래밍 작업이 선행되어야 했으며, 모든 노드에서 추출된 데이터는 일단 베이스스테이션으로 전송된 후 호스트-PC에 저장 및 처리되는 시스템이었다. 하지만 이러한 방법은 미리 프로그래밍 해놓은 방법에 따라 데이터를 추출하기 때문에 유연성이 떨어지고, 모든 노드가 데이터를 베이스스테이션으로 전송해야 하기 때문에 전송비용이 많이 든다. 이러한 단점을 최소화하기 위하여 In-Network 질의 처리 시스템이 필요하게 된다[2]. 또한 일종의 SQL과 유사한 질의가 센서네트워크 내에서 처리됨으로써 어떤 센서 노드로부터, 어떠한 방법으로 데이터를 추출해야 할지를 지시할 수 있게 된다. 따라서 센서네트워크 상에서의 질의처리라는 개념을 적용시켜 좀 더 유연하고 효율적인 시스템을 만들 필요가 있다. 이와 같은 개념을 적용시켜 현재 센서 네트워크 데이터베이스 연구는 COUGAR[3]나 TinyDB[4] 시스템을 중심으로 진행되어오고 있다.

본 논문에서는 유한 상태 머신 기반의 실행모델을 이용하고 응용의 변화에 대처할 수 있게 동적 재구성 기능을 지원하도록 설계된 SenOS[5][6]상에서의 센서 네트워크 데이터베이스 시스템인 SenDB를 설계하고 COUGAR나 TinyDB 시스템의 문제점을 지적함과 동시에 해결책을 살펴보도록 한다.

2. 관련연구

2.1. COUGAR

센서 네트워크 데이터베이스 시스템은 코넬대학의 COUGAR 시스템과 버클리 대학의 TinyDB를 중심으로 연구되어져 왔다. 우선 센서 네트워크용 분산 데이터 처리 시스템인 COUGAR 시스템은 센서노드들이 사전에 프로그래밍 되고 데이터의 수집 및 처리를 중앙에서 처리하는 중앙 집중형 시스템과는 달리 데이터 접근과 처리를 모두 분산된 형태로 처리한다. 이와 같은 접근방법은 SQL과 유사한 센서기반의 질의를 사용하여 사용자가 원하는 다양한 정보를 추출할 수 있을 뿐만 아니라, 센서노드 내에서 데이터가 처리되는 계산비용보다 전송비용이 크다는 특징[7]을 적용하여 분산된 형태의 처리방법으로 전송비용을 절감시킴으로써 센서네트워크 전체의 생명주기를 늘릴 수 있다는 장점을 가지고 있다. 또한

질의 처리를 위하여 연속 질의의 실행 주기를 표현하기 위한 $every()절을 포함하는 SQL과 유사한 질의 언어를 제공하는 특징을 가진 시스템이다[2].

2.2. TinyDB

또 하나의 센서 네트워크 데이터베이스 시스템인 TinyDB는 TinyOS에서 구동되는 센서 네트워크로부터 정보를 얻어내기 위한 질의 시스템으로써 기본적으로 센서 노드들로 구성된 네트워크 환경을 데이터베이스 테이블로 보는 개념을 기반으로 한 시스템이다. 환경(sensor field)에 설치된 노드들로부터 데이터를 모아 필터링하고 취합하여 Host-PC로 전송한다. TinyDB는 또한 COUGAR 시스템과 같이 SQL과 유사한 센서기반 질의 언어인 AQL(Acquisitional Query Language)를 정의하고 있으며, 질의를 효율적으로 처리하기 위한 에너지 효율적인 분산 질의처리 방법을 제안하고 있다[8].

하지만 TinyDB에서 사용자에 의해 정의된 질의를 패킷화하여 전송하게 될 때, 각 속성 및 각 조건표현절 마다 한 개의 45바이트 패킷, 즉 질의 당 다중 패킷을 사용하게 된다. 이러한 패킷구조는 전송비용을 최소한으로 줄여야하는 센서네트워크의 특징에 따라 최적화하여 구성될 필요성이 있다. 또한 COUGAR와 TinyDB 시스템은 기본적으로 동종간의 노드 상에서 데이터를 처리하는 것에 초점이 맞추어져있기 때문에 새로운 센싱장치가 탑재된 센서노드를 사용하기 위해서는 재컴파일과 같은 작업이 필요하게 된다. 따라서 본 논문에서는 센서네트워크 상에서 사용되는 전송비용을 줄이기 위하여 Packet 구조를 재정의 하였으며, 새로운 센싱장치가 탑재된 센서 노드간의 연동문제에 대하여 살펴볼 것이다.

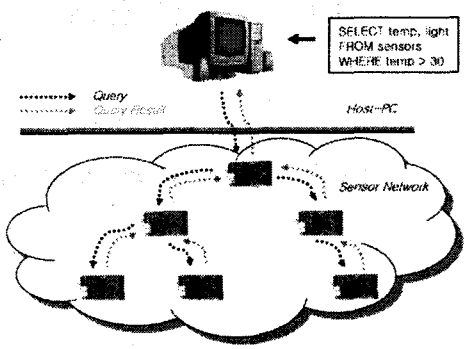
3. SenDB 구조

우선 전체적인 SenDB의 구조는 <그림 1>과 같이 크게 Host-PC 부분과 센서네트워크 부분으로 구성되어있다. Host-PC 부분은 사용자에 의해 입력된 SNQL(Sensor Network Query Language)문의 검사 및 In-Network 질의 처리를 수행하는 노드에게 전송되는 패킷을 생성하는 소프트웨어가 존재하게 된다. 또한 Host-PC는 센서네트워크로부터 전송되어온 질의의 결과를 화면에 출력해 주거나 혹은 저장장치에 저장을 하는 역할을 하게 된다. 센서네트워크 부분은 각 노드에 탑재되어 있는 질의 처리기이다. 즉 Host-PC로부터 전송받은 패킷에 대하여 In-Network 질의 처리를 수행하는 처리기가 센서네트워크 내의 각 노드에 탑재되어 동작하게 된다.

3.1. Sensor Network Query Language(SNQL)

본 논문에서 제안한 SenDB는 센서네트워크상에서 다양한 질의를 표현하기 위하여 Sensor Network Query Language(SNQL)를 정의하였다. SNQL은 기존의 센서네트워크 데이터베이스 시스템에서 사용되어지는 질의어와 유사하게 SQL과 유사한 센서 기반 질의 언어로 표현되어 있으며, 사용자는 SNQL과의 인터페이스를 통해 관심 있는 데이터를 기술할 수 있게 된다.

센서네트워크 상에서 질의어를 처리할 때는 주로 제한된 질의의 처리가 이루어진다. 즉 다시 말해 전통적인 DBMS



<그림 1> SenDB Architecture

에서 질의를 처리할 때는 저장되어 있는 데이터를 사용하여 복잡한 질의를 만들고 처리하는 반면, 센서 네트워크에서는 처리해야 할 질의가 제한되어 사용되기 때문에 전통적인 DBMS상의 질의 Predicate에 비해 비교적 적은 종류의 질의 Predicate가 사용된다.

기본적인 SQL Predicate인 SELECT, FROM, WHERE에 대하여 <표 1>과 같이 정의하였으며, 센서네트워크만의 특징인 연속 질의의 실행 주기를 나타내기 위한 Predicate인 INTERVAL 및 센서 노드의 Actuator 질의의 특징에 따라 추가된 질의 Predicate를 <표 2>와 같이 표현 하였다.

<표 1> 기본적인 질의 Predicate

SELECT	각각의 센서 노드에서 Sensing할 수 있는 기능 중 사용자가 관심 있어 하는 기능을 나타내고 있음. 또한 MIN, MAX, SUM과 같은 Aggregation Query를 표현하기 위하여 Aggregation Clause를 확장하여 사용 할 수 있음. (ex) [SELECT temp] 또는 [SELECT SUM(temp)]
FROM	전체 센서네트워크는 하나의 큰 sensors라는 단위로 정의된다.
WHERE	각 질의에 해당하는 조건을 나타낸다. 예를 들어 WHERE 온도 > 30과 같은 조건을 사용하여 질의에 적용되는 노드의 수를 줄여 에너지 효율을 높일 수 있다.

<표 2> 확장된 질의 Predicate

INTERVAL	사용자가 Host-PC를 통하여 정의한 센서 노드 상에 적용되는 연속질의의 실행주기를 설정해 줄 수 있다. (ex) INTERVAL 60s

TRIGGER ACTION	질의의 결과가 임의의 조건을 만족시킬 때마다 트리거 액션이 수행될 수 있다. 예를 들어 임의의 질의가 센서 노드에서 처리 될 때, 해당 Actuator가 동작할 수 있는 형태로 구성할 수 있다.

3.2. SenDB Packet 구조

사용자로부터 입력된 질의의 내용을 각 센서노드에서 직접 처리하는 것은 전송비용 및 질의어의 분석과 처리에 의한 프로세싱 비용에 큰 영향을 미친다. 따라서 질의어는 각 센서노드에서 분석과정 없이 바로 처리할 수 있는 구조로 변환되어야 하고, 이러한 데이터, 즉 패킷은 센서네트워크의 자원제한적인 특징에 만족해야 함으로, 가능한 최소의 용량을 가질 수 있도록 설계되어야 한다.

우선 TinyDB에서 패킷이 어떻게 생성되고 처리되는지 살펴보면 사용자가 입력한 AQL문을 Host-PC에서 파싱을 하여 구문분석을 수행한다. 이와 동시에 AQL내의 Predicate를 분석하여 TinyDBQuery라는 객체내의 데이터들을 변경함으로써 패킷을 생성할 준비를 하게 된다. 이때 질의마다 유일한 값을 가지는 QueryID, 전송하는 노드를 가르키는 FwdNode, 몇 개의 속성을 사용하는지 판단하는 NumField, 몇 개의 조건구문을 사용하는지 판단하는 NumExpr등의 데이터를 저장하고 패킷을 생성한다. 또한 TinyDB에서는 패킷을 속성의 개수와 조건구문의 수만큼, 즉 하나의 질의에 대하여 여러 패킷이 각 센서 노드로 전송된다.

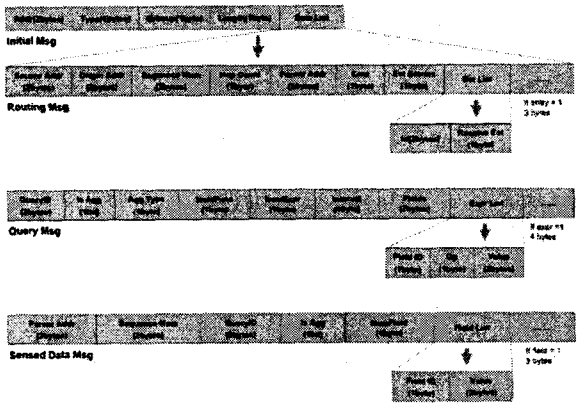
```
SELECT temp, light
FROM sensors
WHERE temp > 30
INTERVAL 60s                    <예제 1>
```

TinyDB에서 <예제 1>과 같은 질의를 전송하기 위해서는 3개의 45바이트 패킷을 전송해야 한다. 또한 처리해야할 속성과 조건구문의 수가 증가 하면, 그만큼 패킷의 수도 증가하기 때문에 전송비용에 매우 큰 영향을 주게 된다. 예를 들어 처리해야할 속성이 습도, 온도, 조도 및 NodeID, 4가지이고 처리해야할 조건구문이 습도, 온도, 조도에 관련하여 3가지 종류가 있다면, 총 7개의 Packet이 각 센서노드로 전송된다. 또한 미리 정해진 45바이트의 패킷이 전송되기 때문에 총 315바이트의 데이터가 전송되는 결과를 초래한다. 반면 우리가 설계한 SenDB에서는 비트연산을 이용하여 2바이트, 즉 16개의 속성을 표현할 수 있도록 설계하였으며, 조건구문의 수에 따라 가변적인 패킷구조를 사용하기 때문에 패킷의 용량을 훨씬 효율적으로 사용할 수 있게 된다. 먼저 이를 위해 SenDB에서 처리해야 하는 질의에 대한 패킷화 작업이 필요하기 때문에 각 질의문의 요소들을 적절히 표현할 수 있는 구조가 필요하게 된다. SenDB에서는 전체 센서 네트워크의 라우팅 경로를 유지하기 위한 Routing Packet, 각 노드에서 센싱된 데이터가 미리 구성된 라우팅 경로에 따라 Host-PC로 전송될 때 사용되는 Sensed Data Packet, 사용자가 원하는 데이터에 대한 질의를 <표 3>에서 정의한 요소의 사용으로 구성되는 Query Packet, 이상 3가지 종류의 Packet을 사용하여 SenDB에서 질의 처리가 이루어진다. 이와 같은 3가지의 Packet은 <그림 2>의 구조로 설계되어 있다. 즉 Initial Packet은 위에서 언급한 3가지의 패킷(Routing, Query, Sensed Data)이 공통으로 사용되는 구조로 구성되어 있다. 다시 말해 전송될 노드의 주소, 패킷의 종류, 총 Packet의 길이 등으로 구성되어 있으며, Routing Packet은 Source Address, Hop Count등으로 구성되어 있다. Host-PC에서 사용자에 의해 정의된 질의를 표현하는 Query Packet은 SenDB에서 정의한 SNQL을 모두 표현 가능한 형태로 구성되어 있으며, 구체적으로 각각의 질의를 식별할 수 있는 Query ID, 집계연산이 필요한 질의인지 판단하는 부분 및 어떤 종류의 집계연산자를 사용하는지 표현해주는 Agg Type, NumField, NumExpr등으로 구성되어 있으며, 가변적인 속성의 수와 조건구문의 수를 처리할 수 있도록 구성되어 있다. 마지막으로 Sensed Data Packet 또한 기본적인 Initial Packet을 사용하며, 부모 주소와 가변적인 속성의 결과값을 표현할 수 있도록 구성되어 있다.

<예제 1>과 같은 SNQL문을 패킷으로 변환하여 처리하였을 때 TinyDB는 총 3개의 Packet이 센서네트워크로 전송

<표 3> Elements in a Query Massage

Element	Description
Query ID	질의에 대한 식별자. 다중질의를 사용할 때, 자식노드로부터 베이스스테이션으로 전송되는 데이터가 어떤 질의에 대한 데이터인지 판별해 주는 기준.
Is Agg	해당 질의가 Aggregation 질의인지 판별해 주는 기준, 즉 질의 내에 Aggregation에 해당하는 구문이 있는지 구분.
Aggregation Type	어떤 Aggregation 구문을 사용하는지 확인. ex) Max = 1, Average = 2 등.
Number of Fields	SELECT 리스트 내에 몇 개의 속성을 사용하는지 확인.
Number of Expressions	WHERE 리스트 내에 몇 개의 조건구문을 사용하는지 확인.
Interval	해당 질의가 얼마의 시간마다 수행되어야 하는지 확인.
Fields	어떤 속성을 사용하고 있는지 표현하는 기준, 예를 들어 2바이트의 길이로 총 16개의 속성을 표현할 수 있음.
Expression Data	각 조건구문의 데이터는 Number of Expressions 값을 참조하여 구조체 형식으로 저장.



<그림 2> SenDB의 Packet 구조

되어지고, 135바이트의 용량을 차지하는데 반해, 우리가 설계한 Query Packet은 하나의 질의 당 한 개의 패킷으로 구성되고, 총 17바이트의 용량을 차지하게 된다. 이는 현재 구현된 TinyDB의 기능을 충분히 살릴 수 있을 뿐만 아니라, TinyDB에 비해 전송비용 또한 줄일 수 있는 장점을 가진다.

3.3. 새로운 센서를 탑재한 노드간의 연동

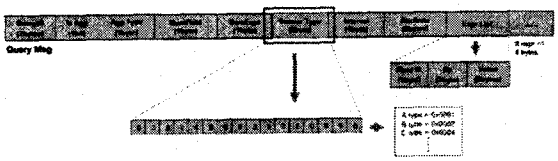
기본적으로 COUGAR와 TinyDB 시스템은 동종간의 노드 상에서 데이터를 처리하는 것에 초점이 맞추어져 설계가 되어 있다. 따라서 새로운 센싱장치(=Attribute)가 탑재된 센서 노드를 사용하기 위해서는 재컴파일과 같은 작업이 필요하게 된다. 하지만 센서 네트워크 데이터베이스 시스템은 새로운 종류의 Sensing장치를 추가한 노드들이 쉽게 네트워크상에 동적으로 추가될 수 있도록 설계되어야한다[9].

SenDB에서는 기존의 시스템에서 고려하지 않은 이종간의 노드 상에서, 즉 새로운 종류의 센싱장치들이 부착된 노드들이 기존에 살포된 노드의 변경 없이 센서 네트워크상에서 동작 가능하도록 설계되었다. 간략하게 설명하자면 각 노드에서 하위노드의 타입정보를 저장하고 있는 필드를 유지함과 동시에 이종 간의 노드를 구별해 줄 수 있는 필드를 <그림 3>과 같이 Query Packet내에 추가하게 된다. 이와 같은 정보를 이용하여 Query Packet이 자식노드로 전송될 때 자신의 노드 상에 저장되어있는 Type정보를 분석함으로써 Query의 전송을 수행할 지 결정하게 된다. 알고리즘은 아래와 같이 정리해 볼 수 있다.

(1) 라우팅 트리를 구성할 때 부모노드들은 해당 자식노드들의 타입정보를 전송 받음으로써 자신의 하위노드 Type이 어떤 것인지 판단할 수 있는 타입정보 필드를 갱신한다.

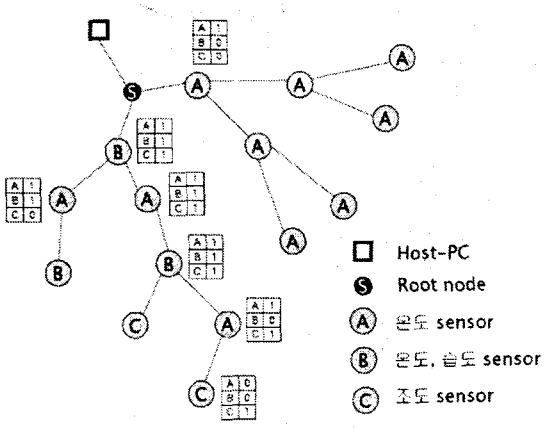
(2) 자식노드가 부모노드로부터 Query Packet을 전달받았을 때 자신의 타입정보 필드를 파악함으로써 질의를 하위노드로 전달해 줄 지 여부를 결정한다.

(3) 새로운 타입의 센서 노드가 추가되었을 때 해당 라우팅 트리가 재구성됨과 동시에 타입정보 또한 갱신됨으로써 (2)번의 동작을 유지할 수 있게 된다.



<그림 3> Sensor Type Field를 추가한 Query Packet

<그림 4>과 같이 부모노드는 자식노드의 타입정보와 함께 해당 필드를 갱신함으로써 사용자에 의한 질의의 전송에 대한 비용을 줄일 수 있다. 예를 들어 조도만을 측정하는 질의를 전송할 때, 만약 부모노드의 타입필드의 노드타입 C필드가 0 이라면 해당 Query Packet을 하위노드로 전송하지 않음으로써 전송비용을 줄일 수 있으며, 새로운 타입의 노드가 추가적으로 살포되면 라우팅 트리를 재구성함과 동시에 타입필드 또한 갱신되기 때문에 재컴파일과 같은 작업 없이 질의처리를 수행할 수 있게 된다. 이와 같은 구성은 TinyDB 혹은 COUGAR 시스템에서 설계되어 있지 않은 이종 간의 센서노드의 구성을 SenDB에서 개선할 수 있다는 장점을 가지고 있다.



<그림 4> 하위노드의 Type 정보 Field를 이용한 이기종간의 효율적인 Query 전송

3.4. SenDB의 설계 및 구현

센서네트워크 운영체제로 제안된 SenOS의 유한 상태 머신 기반의 실행모델을 이용하여 센서 네트워크 상에서 질의를 처리할 수 있는 시스템을 구성해 보았다. SenOS에서 태스크를 처리하기 위해서는 상태모델링, 상태전이테이블 생성, 콜백 함수가 필요하게 된다[5][6]. 따라서 본 논문에서 제안된 시스템을 구축하기 위해서 각 센서노드에서 질의처리를 수행할 수 있는 상태모델링이 필요하고, 이를 상태전이테이블로 구성하는 작업이 요구된다. 또한 상태전이테이블의 액션항목을 참조하여 필요한 콜백 함수를 구성한다. 그 결과는 <그림 5>와 같이 나타낼 수 있다.

Current State	Input event	Destination State	Action
Idle	recv packet	Checkmsg	check recv packet, create data or mhop or query msg
Idle	set interval	Timer	set timer with inter parameter
Timer	set interval complete	Idle	do nothing
Checkmsg	recv query msg	Recvquery	select route
Checkmsg	recv sensed data msg	RecvsensedData	update neighbor info, update routing table or select route
Checkmsg	recv mhop msg	Recvmhop	update neighbor info, update routing table or create new entry
Recvquery	send query msg	Routequery	send query packet to child node
RecvsensedData	ready data	Idle	select route, send recved sensed data packet
Recvmhop	ready data	Idle	do nothing
Idle	timer interrupt	Sense	sensing
Sense	sense complete	Check where	check where predicate
Check where	send sensed data msg	RoutesendedData	get sensed data, select route, send sensed data packet
Check where	drop data	Idle	do nothing
Idle	route query	Routequery	send query packet to child node
Routequery	send complete	Idle	create 'set interval' event
Idle	route sensed data	RoutesensedData	get sensed data, select route, send sensed data packet
RoutesendedData	send complete	Idle	do nothing
Idle	send mhop	Routemhop	choose parent, update routing table, send mhop packet
Routemhop	send complete	Idle	do nothing

<그림 5> 기본적인 질의처리를 위한 상태전이테이블

4. 결론

본 논문에서는 유한상태머신 기반 운영체제인 SenOS상에서 질의를 처리할 수 있는 시스템의 구조에 대하여 상태 전이 테이블과 함께 간략하게 기술하였으며, 기존의 센서네트워크 데이터베이스 시스템의 문제점인 질의 당 다중패킷의 문제, 이종 간의 노드를 센서네트워크상에 적용하지 않은 문제를 제시하고 해결책을 살펴보았다. 향후 본 논문에서 언급하고 있지 않은 TAG[10] 혹은 TiNA[11]와 같은 집계연산 방법이 SenDB상에 적용되었을 때 어떠한 문제점이 제시될 것이며, 또한 기존의 시스템과의 성능 차이에 대하여 살펴볼 것이다. 아울러 교체 가능한 상태전이테이블과 콜백 라이브러리를 이용하여 동적으로 노드를 재구성할 수 있는 SenOS상에서 SenDB의 효율적인 동적 재구성 방법에 대하여 살펴볼 예정이다.

참고 문헌

[1] I.F.Akyildiz, W. Su et al., "A Survey on Sensor Networks", IEEE Communications magazine, August 2002.

[2] Philippe Bonnet, Johannes Gehrke, Praveen Seshadri, "Towards Sensor Database Systems", In Conference on Mobile Data Management, January 2001.

[3] http://www.cs.cornell.edu/database/cougar/

[4] http://berkeley.intel-research.net/tinydb/

[5] Seongsoo Hong, T.-H. Kim, "SenOS: state-driven operating system architecture for dynamic sensor node reconfigurability", International Conference on Ubiquitous Computing, pp.201-203, Oct. 2003.

[6] T.-H. Kim, Seongsoo Hong, "State machine based operating system architecture for wireless sensor networks", Lecture Notes in Computer Science, vol. 3320 no. pp.803, Dec. 2004.

[7] G. J. Pottie, W. J. Kaiser, "Wireless integrated network sensors", Communications of the ACM, 43(5):51-58, May. 2000.

[8] Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, Wei Hong, "TinyDB: An Acquisitional Query Processing System for Sensor Networks", ACM Transactions on Database Systems, Vol. 30, No. 1, March 2005, Pages 122-173.

[9] Johannes Gehrke, Samuel R. Madden, "Query processing in sensor networks", IEEE CS and IEEE ComSoc, Vol. 3, 2004.

[10] Samuel madden, Michael J. Franklin, Joseph Hellerstein, Wei Hong, "TAG: a Tiny Aggregation Service for Ad-Hoc Sensor Networks. In Proceedings of OSDI, 2002.

[11] M. A. Sharaf, J. Beaver, A. Labrinidis, and P. K. Chrysanthis. "TiNA: A Scheme for Temporal Coherency-Aware in-Network Aggregation". In Proceedings of ACM MobiDE Workshop, Sept 2003.