# 이기종 센서 노드 네트워크를 위한 센서용 질의처리 향상

김민규° 김태형
한양대학교 컴퓨터공학과
mgkim°@cse.hanyang.ac.kr, tkim@cse.hanyang.ac.kr

# Improving Sensor Query Processing for Heterogeneous Sensor Networks

Minkyu 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. 이에 따라 Declarative Query Language를 이용하여 다양한 질의를 표현하고, 이와 같은 질의를 효율적으로 처리하기 위한 에너지 분산 질의처리 방법이 중요한 이슈로 부각되고 있다. 본 논문은 [3]의 확장된 논문으로써 유한 상태 머신 기반 운영체제인 SenOS상에서 질의를 처리할 수 있는 시스템의 구조 중 SenOS의 동적 재구성 기능적 특성을 적용한 SenDB의 동적 Aggregation Function 추가 기능에 대하여 살펴보았다. 아울러 [3]에서 제안한 이기종 센서노드 연동기능의 개선점 및 구현 방법에 대하여 살펴보겠다.

## 1. 서론

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 센싱장치들이 물리적 신호, 다시 말해 열, 소리, 압력 등에 반응하여 데이터를 추출해야 할 뿐만 아니라 각각의 노드는 이웃 노드 혹은 베이스스테이션으로 센싱된 데이터를 전송할 수 있어야 한다. 기존의 무선 센서 네트워크상에서 데이터를 추출하는 방법은 중앙 집중형 방식을 주로 사용하였다. 다시 말해 모든 센서 노드들이 개발자에 의해 미리 프로그래밍 되어 다양한 환경에 뿌려지며, 일단 뿌려진 노드들은 프로그래밍 된 동작 이외의 작업은 수행할 수 없는 시스템이었다. 따라서 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 Host-PC에 저장 및 처리되는 시스템이었다. 하지만 이와 같은 방법은 미리 프로그래밍 해놓은 방법에 따라 데이터를 추출하기 때문에 유연성이 떨어지고, 모든 노드가 데이터를 베이스스테이션으로 전송해야 하기 때문에 전송비용이 많이 소모된다. 이러한 문제점을 개선하기 위하여 센서 네트워크용 질의처리 시스템인 TinyDB[2], SenDB가[3] 연구되어지고 있다. 센서 네트워크용 질의처리 시스템에서는 Pre-defined Programming 방법을 개선하기 위하여 사용자에게 일종의 SQL과 유사한 Query를 제공함으로써 어떤 센서 노드로부터, 어떠한 방법으로 데이터를 추출해야 할지를 지시할 수 있게 된다. 따라서 센서 네트워크상에서 질의처리라는 개념을 적용시켜 좀 더 유연하고 효율적인 시스템을 구성할 수 있게 된다.

본 논문에서는 [3]의 확장된 논문으로써 이미 뿌려진 노드상의 동작을 SenOS[4][5]의 특징인 상태전이테이블

의 교체만으로 추가 할 수 있는 방법에 대하여 제안하였으며, 새로운 센서를 탑재한 노드간의 연동문제를 기존 SenDB의 라우팅테이블을 수정하여 개선하고자 한다. 전체적인 구성은 첫째, 센서 네트워크를 위한 질의처리 시스템에 대하여 설명하였고, 2장에서는 SenDB, TinyDB 등의 관련연구에 관하여 설명하였으며, 3장에서는 SenOS의 동적 재구성 방법을 이용한 추가적인 Aggregation 기능에 대한 추가, 4장에서 새로운 센서를 탑재한 노드간의 연동문제에 대한 개선 및 구현방법에 대하여 설명하였다. 마지막으로 5, 6장에서는 실험 및 평과와 결론 및 향후 연구과제에 관하여 설명하였다.

## 2. 관련연구

### 2.1 TinyOS와 TinyDB

센서 네트워크를 위해서 설계된 오픈 소스 임베디드 운영체제인 TinyOS는 nesC 언어에 의해 제공되는 컴포넌트 기반 모델이다. TinyOS는 메모리와 디바이스 관리를 포함하고, 통신 레이어들을 위하여 태스크 스케줄링과 프로토콜 스케줄링을 포함하고 있다. 또한 모든 모트들은 ROM에 200K로 프로그래밍 된 작은 TinyOS[6]를 가지고, 그 위에 애플리케이션을 실행하게 된다.

TinyDB는 TinyOS 위에서 정보를 추출하기 위한 질의처리 시스템이다. TinyDB는 사용자가 정보를 쉽게 추출할 수 있도록 SQL과 유사한 선언적 질의언어를 제공해 주고 있기 때문에 임베디드 C코드와 같은 작성이 불필요하게 된다. 또한 질의를 효율적으로 처리하기 위한 에너지 효율적 분산 질의처리 방법을 제안하고 있다. 이와 같은 기능을 가진 TinyDB는 센서 노드들로부터 데이터를 모아 필터링하고 취합하여 Host-PC로의 효율적인 전달을 목적으로 하고 있다.

### 2.2 SenOS와 SenDB

센서 네트워크 운영체제로 제안된 SenOS는 이벤트 구동 방식으로 동작하고 유한 상태 머신 기반의 실행 모델을 이용하여 설계되었다[4][5]. 센서 네트워크 환경에서 사용되는 무선 센서 노드들은 컴퓨팅 파워와 메모리 등이 제한적이며, 재사용을 고려하지 않은 일회용 컴퓨팅 플랫폼이라는 특징을 가진다. 따라서 이러한 센서 노드들은 저 전력 경량 장치에서 동작할 수 있으면서 동시에 환경과 응용 프로그램의 변화에 대처하기 위해 동적인 재구성을 지원할 수 있는 구조의 운영체제가 필요하다. SenOS는 교체 가능한 상태전이테이블과 Callback 라이브러리를 이용하여 매우 효과적인 동적 노드 재구성을 지원한다는 장점을 가지고 있다. 다시 말해 각 노드의 현재상태와 적용되는 이벤트, 다음상태와 해당 Callback 함수를 사용함으로써 상태전이테이블을 이용한 비즈니스 로직을 구성할 수 있다. 이와 같은 상태전이테이블은 SenOS상에서 하나의 응용프로그램의 역할을 하게 된다. 이와 더불어 해당 상태전이테이블의 업데이트를 이용하여 센서 네트워크의 동적 노드 재구성을 지원하게 된다. [3]에서 언급한 SenDB는 SenOS상에서 센서 네트워크용 질의처리 시스템에 해당하는 상태전이테이블과 관련 Callback 함수를 설계함으로써 효율적인 분산 질의처리 방법을 제안하였으며, TinyDB에서 사용하고 있는 질의당 다중 패킷 문제에 대한 개선점을 제안하였다. 이와 함께 본 논문에서는 새로운 센서를 탑재한 노드간의 연동문제를 각 노드가 유지하고 있는 라우팅 테이블에 Group Field를 추가함으로써 효율적으로 개선 및 구현하고자 하였으며, 이미 뿌려진 노드상의 Aggregation 기능을 SenOS[4][5]의 특징인 상태전이테이블의 교체만으로 추가 할 수 있는 방법에 대하여 제안하고자 한다.

## 3. SenDB상에서의 동적 재구성 기능

### 3.1 SenOS의 동적 재구성 기능

센서 노드의 Resource와 Task를 효과적으로 관리할 수 있는 대표적인 센서 노드용 운영체제인 TinyOS는 센서 노드에 최적화 시킨 운영체체이지만 동적 재구성 기능을 지원하지 않는다. 노드에 프로그래밍된 TinyOS 시스템은 컴파일 시점에 정적으로 링크된 바이너리 이미지이다. 따라서 변경된 응용 프로그램은 재 컴파일 과정을 통해 새롭게 이미지를 생성하고 센서 노드에 유선 통신을 이용하여 프로그램 하여야 하므로 시스템을 실행 중에 동적으로 재구성하는 것이 쉽지 않다. 물론 XNP[7]와 Mate[8]를 사용하여 TinyOS의 동적 재구성 기능을 확장하여 사용한 경우도 있지만, XNP와 같은 경우 새롭게 생성된 전체 시스템 이미지를 메모리에 프로그래밍하고 재부팅을 거치는 과정이 수행되기 때문에 무선통신을 통하여 전체 시스템 이미지를 전송해야 한다는 단점을 가지고 있다. 또한 Mate는 TinyOS 상에서 동작하는 가상 머신이라는 점에서 노드마다 가상 머신이 설치되어야 하는 시스템 자원의 추가적인 소모에 대한 단점을 가지고 있다. 이에 반해 우리가 설계하고 구현한 SenOS상에서는 이러한 동적인 재구성 기능을 상태전이테이블의 교체를 이용하여 구성할 수 있기 때문에 좀 더 효율적인 동적 재구성기능이 가능하다[9].

3.2 SenDB상에서 상태전이테이블의 Update를 이용한 동적 재구성

TinyDB에서는 새로운 Sensors, Control/Actuation, Data Processing Logic, Events 등의 확장을 위해서 반드시 오프라인에서 해당 Component를 추가한 후 재 컴파일 해야 하는 과정이 필요하다. 반면 SenOS 상에서는 상태전이테이블의 변경을 이용하여 동적 재구성 방법을 제공하고 있다. 그 중 SenOS에서 제공해 주고 있는 System Callback 함수와 특정한 응용프로그램을 위한 최소한의 User Callback 함수가 프로그래밍 되어 있을 때, 상태전이테이블의 교체만으로 응용프로그램의 동적 재구성이 수행될 수 있다. 예를 들어 SenDB에서 Aggregation과 같은 연산은 SenOS에서 제공해 주는 System Callback 함수와 최소한의 SenDB Callback 함수를 사용하여 Data Processing Logic을 바꾸어 줄 수 있다. 즉 상태전이테이블의 Update 만으로 응용프로그램을 재구성 시킬 수 있게 된다. 우선 Aggregation 연산을 수행하기 위하여 System이 제공해야 할 Callback 함수는 <그림 1>에서 나타난 것과 같이 정의 할 수 있다.

| SenOS System Callback 함수 | Description |
|---|---|
| void plus (void *param); | + 연산을 수행하는 Callback함수 |
| void minus (void *param); | - 연산을 수행하는 Callback함수 |
| void multiple (void *param); | × 연산을 수행하는 Callback함수 |
| void divide (void *param); | / 연산을 수행하는 Callback함수 |
| void compareBigger (void *param); | < 비교 연산을 수행하는 Callback함수 |
| void compareSmaller (void *param); | > 비교 연산을 수행하는 Callback함수 |
| void increase (void *param); | 증가 연산을 수행하는 Callback함수 |
| void decrease (void *param); | 감소 연산을 수행하는 Callback함수 |

그림 1 SenOS에서 제공하는 System Callback 함수

Aggregation에서 사용하는 데이터는 자식노드로부터 전송받은 SensedData와 현재 노드에서 센싱된 CurrentData가 필요하다. 이러한 데이터의 처리는 상태전이테이블 만으로 처리하기 힘들기 때문에 SenDB용 Callback 함수인 sdbSetSensedData (void *param)와 sdbSetCurrentData (void *param) 함수가 필요하게 된다. 또한 PLUS 혹은 MINUS와 같은 System Callback 함수로부터 도출된 결과데이터를 Current Sensed Data에 저장하는 동시에 또 다른 자식노드의 Sensed Data가 있는지 검사하는 sdbStoreAggCheckData (void *param) 함수가 필요하게 된다. 이와 같은 SenDB용 Callback 함수를 사용하여 또 다른 User Defined Aggregation 기능에 적용할 수 있게 되므로, 기본적인 SenDB용 Callback 함수가 센서노드에 올라가 있을 때, 상태전이테이블의 교체만으로 동적 재구성이 가능하게 된다. 자식노드로부터 전송된 Sensed Data와 현재 노드에서 센싱된 Current Data를 비교하여 더 큰 값을 부모노드로 전송하는 MAX와 같은 Aggregation 기능은 <그림 2>와 같은 상태 모델로 구성할 수 있다. 이후 MAX에 대한 상태 모델을 사용하여 <그림 3>와 같은 상태전이테이블로 전환하여 사용할 수 있게 된다. 이후 추가적인 Aggregation 기능이 필요해 진다면, 예를 들어 SUM과 같은 Data Processing Logic이 필요하게 되면 <그림 4>, <그림 5>과 같은 상태 모델, 상태전이테이블을 사용하여 동적으로 응용프로그램을 재구성 시킬 수 있게 된다.

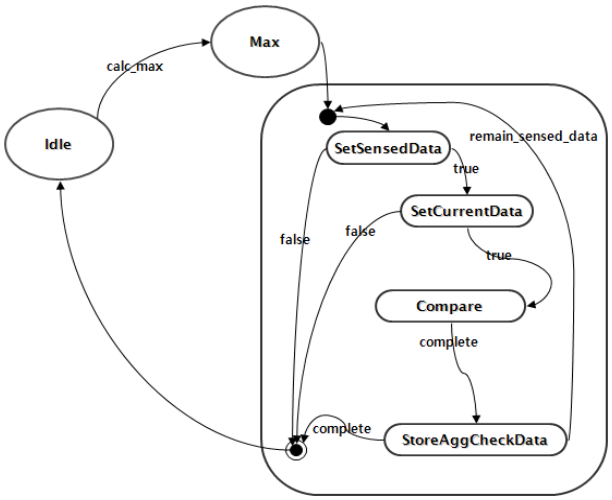


그림 2 Aggregation MAX 상태 모델

| 현재상태 | 이벤트 | 다음상태 | Callback함수 |
|---|---|---|---|
| Idle | calc_max | SetSensedData | sdbSetSensedData |
| SetSensedData | true | SetCurrentData | sdbSetCurrentData |
| SetCurrentData | true | Compare | compareBigger |
| Compare | complete | StoreAggCheckData | sdbStoreAggCheckData |
| SetSensedData | false | Idle | doNothing |
| SetCurrentData | false | Idle | doNothing |
| StoreAggCheckData | remain_sensed_data | SetSensedData | sdbSetSensedData |
| StoreAggCheckData | complete | Idle | doNothing |

그림 3 Aggregation MAX 상태전이테이블

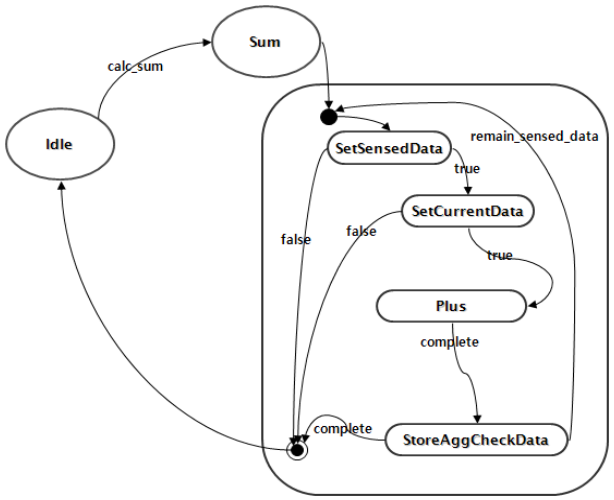


그림 4 Aggregation SUM 상태 모델

| 현재상태 | 이벤트 | 다음상태 | Callback함수 |
|---|---|---|---|
| Idle | calc_sum | SetSensedData | sdbSetSensedData |
| SetSensedData | true | SetCurrentData | sdbSetCurrentData |
| SetCurrentData | true | Plus | plus |
| Plus | complete | StoreAggCheckData | sdbStoreAggCheckData |
| SetSensedData | false | Idle | doNothing |
| SetCurrentData | false | Idle | doNothing |
| StoreAggCheckData | remain_sensed_data | SetSensedData | sdbSetSensedData |
| StoreAggCheckData | complete | Idle | doNothing |

그림 5 Aggregation SUM 상태전이테이블

이와 같은 SenDB용 확장 Callback 함수와 해당 상태 전이테이블을 이용하여 동적 재구성이 가능한 구조에 대하여 Aggregation 기능을 예로 들어 살펴보았다. 이와 같은 SenOS의 동적 재구성기능을 SenDB에 적용하여 기존의 TinyDB에서 제공하지 않는 동적 Aggregation Function 추가기능을 제공해 줄 수 있다는 장점을 가지고 있다.

## 4. 이기종 센서노드간의 연동기능에 대한 개선 및 구현

### 4.1 TinyDB에서의 이기종 센서노드간의 연동 기능

TinyDB 시스템은 센서 네트워크에서 받아진 센서의 종류를 기술하기 위해 메타데이터 카탈로그를 제공하고 있다. 다시 말해 특정 센서를 적용시킨 Query가 각 노드에 전송되었을 때, 노드내의 메타데이터 카탈로그를 분석하여 해당 Query의 적용 유무를 판단하게 된다. 하지만 메타데이터 카탈로그를 사용하는 TinyDB 시스템에서는 새로운 센서가 탑재된 노드가 이미 구성된 센서 네트워크상에 살포되고 새롭게 추가된 Attribute를 Host-PC에서 Query에 적용 할 경우, 해당 센서를 탑재하지 않은 노드들은 NULL값, 즉 쓸모없는 데이터를 Host-PC로 전송하게 됨으로써 전송비용이 커지게 된다. 또한 사용자가 원하는 Attribute값이 포함된 Query는 미리 생성된 전체 센서 네트워크 라우팅 트리를 사용하기 때문에 라우팅 트리 상의 모든 센서노드가 해당 질의에 반응하여야 한다. SenDB에서는 이와 같은 이기종 센서노드간의 Query 결과 데이터 전송을 각 노드가 유지하고 있는 라우팅 테이블에 Group Field를 추가함으로써 효율적으로 개선 및 구현하고자 하였다.

### 4.2 SenDB에서의 이기종 센서노드간의 연동기능 개선 및 구현

SenDB에서는 이기종 센서노드의 연동을 위해 메타데이터 카탈로그를 사용하는 TinyDB의 문제점을 개선하기 위하여 라우팅 테이블과 Query Packet에 그룹 필드를 추가하여 설계 및 구현하였다. SenDB에서 사용하는 라우팅 트리는 싱크 노드로부터 가장 적은 홉 카운트를 가지고 최적으로 평가된 링크 Cost 값을 갖는 노드를 부모노드로 선택함으로써 생성된다. TinyDB 시스템에서 사용된 전체 컴포넌트에서 실질적으로 라우팅을 수행하는 기능을 담당하고 있는 컴포넌트(EWMAMultihopRouter)만을 분리시켜 SenOS에서 동작하는 SenDB 시스템으로 재설정하였다. 이기종 센서노드간의 연동기능은 [3]에서 제안된 알고리즘을 이용하였으며, 우선 <그림 6>과 같이 TinyDB 시스템에서 사용하는 라우팅 테이블에 Sensor Type 필드를 추가하여 각 노드가 유지해야 하는 하위노드의 타입정보 필드를 저장하게 된다.

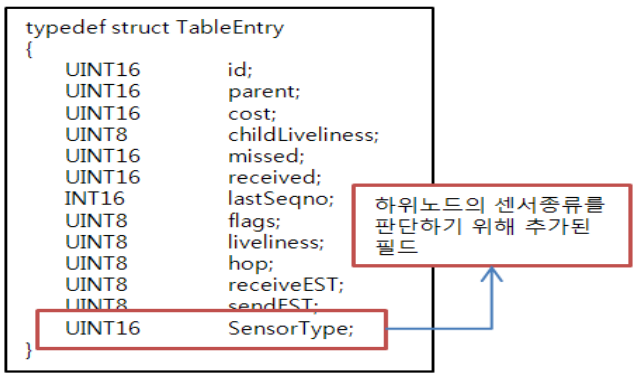


그림 6 Sensor Type 필드를 추가한 라우팅테이블

각각의 센서노드들은 라우팅을 위한 메시지를 20초 간격으로 발생시키게 되고, 해당 메시지를 전송받은 상위 노드들은 메시지 내에 저장된 하위노드의 정보를 이용하여 라우팅테이블을 갱신하게 된다. 이와 더불어 새로운 타입의 센서 노드가 추가되었을 경우 해당 라우팅 트리가 재구성됨과 동시에 타입정보 또한 갱신됨으로써 센서 네트워크내의 모든 노드들은 하위노드들의 정보를 유지할 수 있게 된다. 이와 같이 각각의 센서노드내에 저장된 하위노드의 타입정보 필드와 Host-PC에서 생성된 Query내에 포함된 타입정보와 AND 비트연산이후 Query를 하위노드로 전달해 줄 지 여부를 결정하는 CompareType SenDB 확장 Callback 함수를 이용하여 Attribute종류에 따른 필터링이 가능하게 된다. 이와 같은 설계와 구현은 TinyDB에서 사용되고 있는 메타데이터 카탈로그의 문제점인 NULL과 같은 쓸모없는 데이터의 전송 및 미리 생성된 라우팅 트리내의 모든 센서노드들이 Query에 반응해야 하는 점을 개선할 수 있다는 장점을 가지고 있다.

## 5. 실험 및 평가

실험은 TinyDB에서 사용되는 Aggregation 방법인 TAG 알고리즘을 SenDB에 적용시켜 Aggregation Query 결과에 대한 정확도를 측정하고 TinyDB와 비교 및 평가한다. 실험을 위해 사용된 센서 네트워크용 노드는 CC2420 RF 모듈과 ATmega128L 프로세서가 장착된 노드들을 이용하여 실험을 수행한다. SenDB의 라우팅 트리 생성 부분은 TinyDB에서 사용되었던 Reliable Routing Protocol을 사용한다. 원하는 데이터에 대한 Query표현 및 Query Result의 전송 및 처리는 Pipelined Aggregate[14] 방법을 사용한다.

<그림 7>와 같이 1개의 Sink 노드와 5개의 중간노드, 총 6개의 노드를 사용하여 [예제]와 같은 Aggregation Query (모든 센서 노드로부터 2초 마다 한 번씩 온도에 대하여 집계 연산된 데이터를 Host-PC로 전송)를 처리하게 된다. 이와 같은 Aggregation Query를 TAG[14] 알고리즘, 다시 말해 In-Network Query Processing을 사용하여 처리하기 위해서는 각 레벨 당 처리하는 노드들의 시간을 [전체시간(INTERVAL) / Tree]의 깊이(Depth)로 나누어 처리하였다.

실험의 결과 [예제]와 같은 Aggregation Query를 TinyDB와 SenDB에서 수행해 본 결과, TinyDB에서는 63.2%, SenDB에서는 69%의 Query Result 정확도가 나타남을 알 수 있었다.

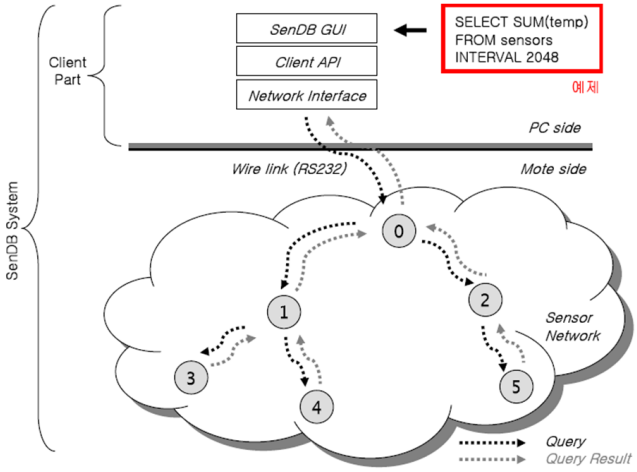


그림 7 Aggregation Query의 정확도 평가를 위한 실험환경

TinyDB에서는 1개의 Query당 다중 패킷을 사용하는데 반하여 SenDB에서는 1개의 Query당 1개의 패킷을 사용한다[3]. 센서 네트워크의 토폴로지가 무선 RF의 특징으로 인하여 불안정함에 따라 라우팅 패킷뿐만 아니라 Query 패킷 또한 주기적으로 전송해 주어야 한다. 이에 따라 TinyDB에서는 Query의 복잡도에 따라 전송되는 패킷의 용량도 커지게 된다. 이는 INTERVAL의 간격이 줄어들면 들수록 각 노드에서 결과 데이터의 연산 및 전송에 필요한 시간 간격이 줄어들기 때문에 집계 연산에 필요한 하위노드에서 상위 노드로의 시간 간격을 보장할 수 없게 된다. 따라서 Query 패킷 및 Query Result 패킷을 SenDB에 최적화 시켜 설계된 SenDB 시스템이 좀 더 높은 정확도를 가질 수 있다.

## 6. 결론 및 향후 과제

본 논문에서는 유한 상태 머신 기반 운영체제인 SenOS상에서 질의를 처리할 수 있는 시스템의 구조 중 SenOS의 동적 재구성 기능적 특성을 적용한 SenDB의 동적 Aggregation Function 추가 기능에 대하여 살펴보았다. 아울러 [3]에서 제안한 이기종 센서노드 연동기능의 개선점 및 구현 방법에 대하여 살펴보았으며, [3]에서 제안된 방법으로 설계 및 구현한 SenDB를 기본적인 Aggregation Query에 초점을 맞춰 실험 및 평가를 해보았다. 본 논문에서 살펴본 것과 같이 SenDB에서는 TinyDB에서 제공하지 않는 동적 Aggregation Function 추가기능을 제공해 줄 수 있으며, 아울러 TinyDB에서 사용되고 있는 메타데이터 카탈로그의 문제점인 NULL

과 같은 쓸모없는 데이터의 전송 및 미리 생성된 라우팅 트리내의 모든 센서노드들이 Query에 반응해야 하는 점을 개선할 수 있다는 장점을 가지고 있다.

향후 개선된 방법을 추가하여 구현된 SenDB 질의 처리 시스템과 중앙 집중형 및 TinyDB 시스템과의 성능비교를 통해 센서 네트워크상의 데이터 처리 효율을 평가할 예정이다. 아울러 교체 가능한 상태전이테이블을 이용하여 동적으로 노드를 재구성할 수 있는 SenOS상에서 SenDB의 효율적인 동적 재구성 방법을 다양한 기능에 대한 비즈니스 로직을 고려하여 실험 및 평가 할 예정이다.

## 참 고 문 헌


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

[2]. Samuel R. Madden, Michael J. Franklin, Josephm. 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.

[3]. 김민규, 김도혁, 김태형, “SenDB: 무선 센서 네트워크용 질의 처리 시스템”, 한국정보과학회 가을 학술발표논문집 Vol. 33, No. 2(A), 2006년

[4]. 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.

[5]. 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.

[6]. TinyOS web site. Available in: http://www.tinyos.net

[7]. Crossbow Technology, Inc. Mote In-Network Programming User Reference, 2003

[8]. P.Levis and D.Culler. "Mate: A tiny virtual machine for sensor networks.", In international Conference on Architectural Support for Programming-Languages and Operating Systems, San Jose, CA, USA, Oct. 2002.

[9]. 김도혁, 김민규, 김태형, “센서 네트워크용 운영체제 SenOS에서 동적 재구성 기능 구현”, 한국정보과학회 2006 가을 학술발표논문집(A), pp.390-394, 2006년 10월.