데이타 기반 센서 네트워크에서 다차원 영역 질의를 위한 동적 데이타 분산

(Dynamic Data Distribution for Multi-dimensional Range Queries in Data-Centric Sensor Networks)

임 용 훈 † 정 연 돈 †† 김 명 호 †††
(Yong Hun Lim) (Yon Dohn Chung) (Myoung Ho Kim)

요 약 센서 네트워크는 온도, 습도 등 서로 연관된 여러 종류의 스칼라 데이타를 감지하기 때문에, 다차원 영역 질의가 유용하게 사용된다. 그리고 데이타 기반 센서 네트워크에서는 데이타가 센서에 직접 저장되기 때문에, 다차원 데이타를 효율적으로 관리하기 위해서는 데이타 주소 지정이 매우 중요하다. 이전의 다차원 영역 질의 처리 기법들은 데이타를 효율적으로 관리하는데 집중하여, 네트워크의 동작 시간(수명)을 고려하지 않았다. 본 논문은 Hilbert 곡선을 이용하여 센서 노드들을 선형화하고, 각 센서에게 데이타 공간을 균일하게 배분시키는, 동적인 데이타 분산 기법을 사용함으로써 네트워크 동작 시간을 연장시키는 방법을 제안한다.

키워드 : 센서 네트워크, 데이타 기반 저장방식, 다차원 영역 질의, 데이타 분산

Abstract In data-centric networks, various data items, such as temperature, humidity, etc. are sensed and stored in sensor nodes. As these attributes are mostly scalar values and inter-related, multi-dimensional range queries are useful. To process multi-dimensional range queries efficiently in data-centric storage, data addressing is essential. The previous work focused on efficient query processing without considering overall network lifetime. To prolong network lifetime and support multi-dimensional range queries, we propose a dynamic data distribution method for multi-dimensional data, where data space is divided into equal-sized regions and linearized by using Hilbert space filling curve.

Key words : sensor network, data-centric storage, multi-dimensional range queries, data distribution

1. 서 론

센서 네트워크는 사막, 심해와 같이 인간이 접근하기 힘든 환경에 컴퓨팅 능력(computing power)을 지닌 초소형의 센서를 배치하여 데이타를 수집 및 처리하고 데이타를 무선으로 전송한다. 센서는 온도, 습도, 진동 등과 같은 여러 가지 값들을 측정하며, 측정된 값은 센서 네트워크에 저장된다. 이렇게 저장된 데이타들은 점 질의(point query) 뿐만 아니라 다차원 영역 질의(multi-dimensional range query)를 사용하여 분석, 처리할 수 있다. 예를 들어 습지에서 자라는 미생물을 관찰하는 센서 네트워크라면 "특정 지역에서 온도 20~30°C, 습도 30~50%일 때의 미생물 번식 상황을 측정하라"는 식의 질의를 효율적으로 처리할 수 있어야 한다. (이 질의는 '온도'와 '습도'라는 두 차원의 속성 값에 대한 영역을 검색 요건으로 사용하는 2차원 영역 질의가 된다.)

센서 네트워크(sensor network)는 일반적인 무선 네트워크와는 다른 여러 가지 특성을 가지고 있다. 첫째, 센서 네트워크는 계산 능력, 저장 공간, 전력 등의 자원(resource)이 지극히 제한적이다. 특히 고갈된 전력을 재충전하거나 배터리를 교환하는 것이 거의 불가능하기 때문에, 센서의 활동 중 가장 많은 전력을 필요로 하는 데이타의 무선 전송을 줄이는 것이 매우 중요하다. 둘째, 각각의 센서는 수동적으로 배치되거나 설정될 수 없

· 본 연구는 한국과학재단 특정기초연구(R01-2003-000-10627-0)의 지원으로 수행되었음
† 정 회 원 : 삼성전자 기술총괄 시스템 연구소
yonghun.lim@samsung.com
†† 종신회원 : 고려대학교 컴퓨터학과 교수
ydchung@korea.ac.kr
(Corresponding author임)
††† 종신회원 : KAIST 전산학과 교수
mhkim@dbserver.kaist.ac.kr
논문접수 : 2005년 4월 28일
심사완료 : 2005년 10월 27일

으며 무작위적으로 배포된다. 그리고 각 센서는 전체 네트워크의 상황을 알 수 없기 때문에 지역적인 정보만으로 네트워크를 구성해야 한다. 셋째, 한 번 배포된 센서는 재사용되지 않는다. 따라서, 센서 네트워크의 동작 시간을 최대한 연장할 수 있는 데이타 처리 방법이 필수적이다. 센서 네트워크의 동작 시간은 가장 에너지 소모가 많은 몇몇 센서의 동작 시간에 의해 결정되기 때문에, 네트워크 내의 모든 센서가 균일하게 전력을 소모하도록 하는 것이 동작 시간을 연장시키는 방법이 될 수 있다.

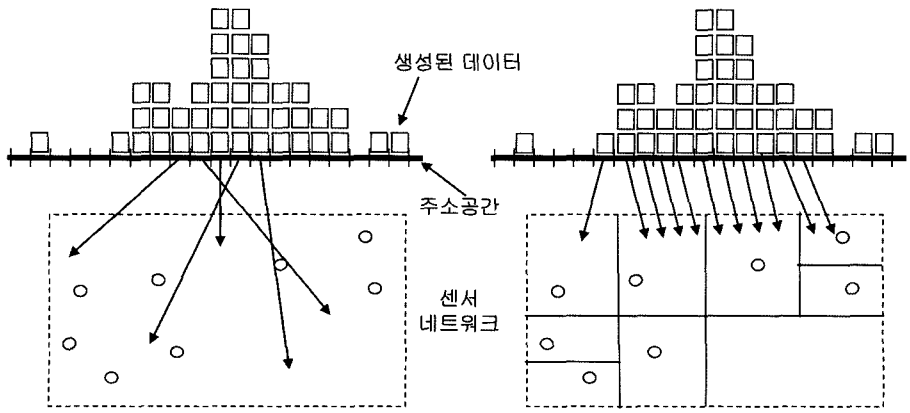
본 논문은 데이타 기반 저장 방식(data-centric storage)의 센서 네트워크를 기반으로 한다. 데이타 기반 저장 방식의 센서 네트워크에서는 생성되는 데이타가 생성된 위치의 센서에 그대로 저장되거나 정해진 외부의 스트림 처리기(stream processor)로 전달되는 것이 아니라, 데이타의 값에 따라 분류되어 특정 센서에 저장된다. 이 방식은 데이타가 많이 발생했을 때 외부 저장 공간에 가까운 센서들에게 부하가 집중되거나, 질의를 수행할 때 필요치 않은 센서가 질의 처리에 관여되는 것을 방지할 수 있다. 본 논문은 센서들이 고르게 전력을 소모할 수 있도록, 모든 센서에게 동일한 데이타 공간을 할당하고, 동적으로 데이타 영역을 조정하여 다차원 영역 질의 시 에너지 사용을 효율적으로 분산시키는 방법을 제안한다. 데이타 공간(data space)은 Hilbert 곡선을 사용하여 주소 지정(addressing)되며, 불균일하게 분포되는 데이타와 질의에 효과적으로 대응하기 위해 동적 데이타 분산(dynamic data distribution) 기법을 적용한다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 관련 연구와 이전의 데이타 주소 지정 방식의 문제점에 대해서 알아보고, 3장에서는 본 논문에서 제시하는 데이타 주소 지정 방식에 대해서 살펴본다. 4장에서는 부하 분산을 위해 사용하는 데이타 분산 기법을 제안하고, 5장에서는 본 논문의 주소 기반 라우팅 방법을 기술한다. 그리고, 6장에서는 시뮬레이션(simulation)을 통해 제안된 기법의 성능을 평가하고, 7장에서 결론과 향후 과제에 대해 서술한다.

2. 관련 연구

데이타 기반 저장 방식에서는 데이타가 저장될 센서를 결정하기 위한 주소 지정 방법이 필요하다. 주소(address)는 데이타가 저장되는 논리적 위치이며, 데이타와 질의를 목적 센서로 라우팅할 때 사용된다(색인(index)이라고 불리기도 한다). 기존의 데이타 기반 저장 방식 센서 네트워크에서는 P2P에서 사용되는 DHT와 유사한 방식인 GHT(Geographic Hash Table) 색인 기법을 사용하여 데이타 저장 위치를 결정하였다[1]. GHT는 데이타의 값을 기초로 지리적 위치(geographic location)를 생성하여, 그 위치에 가장 가까운 센서에 데이타를 저장하는 방식이다. 이 방법은 정합 질의(exact match query)나 접두사 질의(prefix query)는 효율적으로 수행할 수 있으나, 영역 질의(range query)를 처리하는데 있어서는 매우 비효율적이다. GHT는 유사한 값을 지리적으로 멀리 떨어진 위치에 저장하기 때문에, 영역 질의를 처리하기 위해서는 네트워크 전역에 흩어진 여러 센서들에게 부분 질의를 전송해야 하기 때문이다.

DIM(Distributed Index for Multi-dimensional data)[2]은 배치되는 센서의 위치에 따라서 센서 네트워크 영역의 x좌표와 y좌표를 번갈아 가면서 나누어, 하나의 영역에 하나의 센서만 남게 될 때까지 영역을 분할하고,



(a) GHT (b) DIM

그림 1 GHT와 DIM의 분산 특성

그 영역에 위치한 센서는 그 영역에 해당하는 주소를 갖는 데이타를 모두 저장하게 된다. 그런데, 센서는 무작위적으로 배치되기 때문에 센서의 분포에 따라서 센서마다 담당하는 영역의 크기가 불균일해진다. 확률적으로 보면, 센서는 자신이 담당하는 데이타 영역의 크기만큼 데이타를 저장하고 질의를 처리하기 때문에, 센서의 전력 소모 역시 센서의 데이타 영역에 비례하게 된다. 다시 말하면, 센서 네트워크의 동작 시간을 늘리기 위해서는 센서들이 고르게 전력을 소모해야 하므로 모든 센서가 비슷한 크기의 데이타 영역을 담당하도록 해야 한다.

DIM은 영역 질의를 효율적으로 처리하기 위해 유사한 값을 가진 데이타가 동일한 혹은 지리적으로 인접한 센서에 저장되도록 한다. 하지만, 비슷한 값을 갖는 데이타를 하나의 센서에 저장하게 되면 데이타가 일부 영역에 집중될 경우에는 그림 1(b)에서와 같이 몇몇 센서가 불균일하게 많은 데이타를 저장하게 되고, 빠른 전력 소모로 인해 네트워크의 수명을 단축시키는 결과를 가져온다.

DIM, GHT처럼 센서의 위치를 기반으로 동작하는 데이타 기반 저장 방식의 센서 네트워크에서는 GPSR[3]을 사용하여 데이타를 라우팅한다. GPSR은 모든 센서들이 자기 이웃 센서들의 위치를 알고 있다고 가정하고, 그리디 포워딩(greedy forwarding)과 퍼리미터 라우팅(perimeter routing)의 두 가지 모드를 사용하여 데이타를 목적지와 가장 가까운 위치의 센서에게 전달하는 방법이다. 그리디 포워딩은 이웃 센서들 중에 현재 센서보다 목적지에 가까운 센서들이 존재할 경우, 그 중 목적지와 가장 가까운 센서에게 데이타를 전달한다. 장애물 혹은 센서가 존재하지 않는 공 지역(void)이 발생할 경우, 이 지역을 반 시계 방향으로 우회하는 퍼리미터 포워딩을 사용하여 목적지 센서에 다다른다.

3. 다차원 데이타의 주소 지정

3.1 센서 네트워크의 선형화

센서 네트워크는 센서가 배치되는 환경의 특성상 장애물이 존재하고 센서의 자원 제약 역시 심하기 때문에 네트워크를 구성할 때 환경에 대한 세심한 고려가 필요하다. 본 논문에서 가정하고 있는 센서 네트워크 환경은 다음과 같다.

- 모든 센서는 동일한 무선 반경을 가진다. 모든 센서는 무선 통신 반경이 정해져 있기 때문에, 반경 외의 센서로 데이타를 전송하기 위해서는 다중-홉 통신(multi-hop communication)을 사용한다.
- 센서는 무선 통신 반경 내에 있는 이웃 센서의 목록을 유지하며 이웃 센서의 최신 정보를 유지한다. 센서는 최신 라우팅 정보를 유지하기 위해서 데이타를 분산할 때마다 이웃 센서에게 그 사실을 전달한다.
- 센서는 장애물 혹은 기능상의 문제로 자신의 위치를 획득하지 못할 수도 있다.

DIM과 같이 지리적 위치를 데이타 주소로 사용하게 되면, 센서의 분포 형태에 따라 각 센서에게 할당되는 데이타 영역이 불균등해진다. 즉, 센서가 고르지 못하게 분포되면 밀도가 작은 지역의 센서는 다른 지역의 센서보다 많은 데이타 영역을 담당해야 한다. 또한, 모든 센서가 고정된 주소 지정 알고리즘을 사용하므로, 센서가 자신의 데이타를 다른 센서로 이양할 수 없다. 예를 들어 그림 2에서 센서 a가 데이타 D_1과 D_2를 담당한다고 할 때, 센서 a가 전력 소모를 줄이기 위해 주소 <100, 60> 영역을 센서 b로, 주소 <100, 50> 영역을 센서 c로 이양하더라도, 다른 센서들은 고정된 주소지정 알고리즘에 따라 이 영역에 대한 데이타를 여전히 센서 a로 보내게 된다. 이를 방지하기 위해서는 데이타 영역 변동 사실을 다른 모든 센서에게 공지하여야 하며, 모든 센서들은 전체 라우팅 테이블을 유지하여야 한다. 또한, 각 센서는 자신의 위치를 결정하기 위해서 GPS를 사용하는데, 일회용 기기인 센서에 GPS 수신기를 장착하는 것은 많은 비용적 부담을 야기하며, 환경에 따라서 수신기가 오동작할 수도 있기 때문에 센서 네트워크 환경에는 부적합하다. 따라서, 본 논문은 데이타의 주소 지정에 있어서 센서의 위치를 배제하고, 균일하게 데이타 영역을 할당하기 위해서 센서 네트워크를 개념적으로 선형화(linearization)하는 방법을 제안한다.

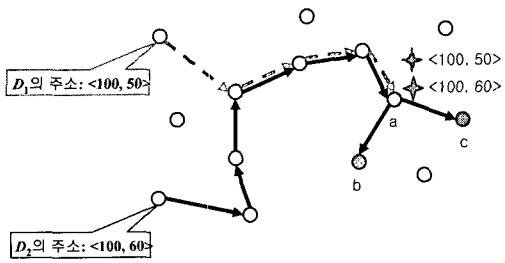
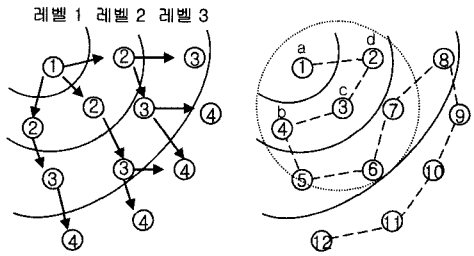


그림 2 위치 기반 주소 지정 기법의 문제점

센서 네트워크의 선형화는 네트워크의 가장자리에 위치한 하나의 센서를 시작점으로 하여 지그재그(zigzag)로 순서를 할당함으로써 이루어진다. 이 형태는 센서에게 주소 영역을 할당할 때 지리적으로 인접한 센서가 인접한 주소 영역을 가질 수 있도록 하며, 데이타 라우팅시 방향성도 제공해 줄 수 있다(라우팅 기법은 5장에서 자세히 설명한다). 센서들은 자신의 위치 정보를 가지고 있지 않기 때문에, 센서 네트워크를 지그재그 형태로 결정짓기 위해서 다음의 단계를 거친다. 먼저 그림

3(a)에서와 같이 시작점 센서는 플러딩(flooding)을 통해 스패닝 트리(spanning tree)를 구성하여 모든 센서들에게 레벨을 부여한다[2-5]. 레벨은 지그재그 형태를 만들기 위한 윤곽을 제시한다. 시작점 센서는 자신에게 번호 1을 부여한 후, 다음 센서를 결정하기 위해 아래와 같은 절차를 따른다.

1. 이웃 센서 중 레벨이 가장 낮은 센서를 먼저 선택한다.
2. 1의 조건이 같은 센서(레벨 n)가 여러 개 존재할 경우, 그 중 레벨 n인 이웃 센서의 수가 가장 적은 이웃 센서를 선택한다. 이는 동일한 조건 하에서는 네트워크의 가장자리에 속한 센서를 먼저 선택하기 위한 휴리스틱이다.
3. 1, 2의 조건이 모두 동일하면 거리가 가장 가까운 (전파 수신 강도가 가장 높은) 이웃 센서를 선택한다.
4. 더 이상 선택할 수 있는 이웃 센서가 없는 경우, 선택 경로를 역행하면서 아직까지 선택되지 않은 센서가 있는지 탐색한다. 센서 번호가 결정되지 않은 이웃 센서를 만나게 될 경우 절차 1~3을 다시 수행한다. 센서 선형화 과정은 센서 1로 돌아왔을 때 종료된다.



(a) 센서 레벨 설정 (b) 센서 선형화

그림 3 센서 네트워크의 선형화

그림 3(b)에서 센서 a는 동일한 레벨(레벨 2)의 이웃 센서가 여럿(센서 b, c, d) 존재하므로 조건 2를 적용한다. 여기서, 센서 3은 동일한 레벨(레벨 2)의 이웃 센서를 두 개(센서 b, d) 가지지만 센서 b와 d는 레벨이 동일한 이웃 센서가 센서 c 하나 뿐이다. 그러므로 센서 a는 센서 b와 d중에서 절차 3에 의해 거리가 가까운 센서 d를 다음 센서로 선택하게 된다.

3.2 데이타 영역의 할당

다차원 데이타 색인 방법으로 공간 채움 곡선(space filling curve)이 있다. 공간 채움 곡선은 균등한 크기의 격자(grid)로 나누어진 다차원 공간에 주소를 할당한다. 하나의 격자 안에 속하는 모든 데이타는 같은 주소를 갖는다. 주소는 일차원 스칼라 값이며, 일차원 주소는 국지적인 데이타 분산에 유용하게 사용될 수 있다(라우팅 테이블을 유지하거나 네트워크 전체에 알리지 않고 동적으로 데이타 영역을 조정하기에 적합하다). 대표적인 방식으로 Z-순서 공간 채움 곡선(Z-order space filling curve), Peano 공간 채움 곡선(Peano space filling curve), Hilbert 공간 채움 곡선(Hilbert space filling curve) 등이 있으며 이 중에서 Hilbert 공간 채움 곡선(이하 Hilbert 곡선)이 지역성을 가장 잘 보존해 주는 것으로 알려져 있다[6]. 지역성은 영역 질의 수행 시 생성되는 부분 질의의 개수에 영향을 미치기 때문에 중요하다. Hilbert 곡선 생성방식은 귀납적(recursive)방식과 상태 다이어그램(state diagram) 방식이 있으며, 상태 다이어그램을 사용하면 적은 저장공간으로 Hilbert 곡선을 생성할 수 있다[7].

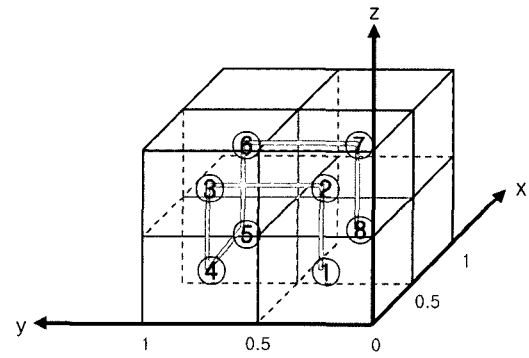


그림 4 차수(Order) 1의 3차원 Hilbert 공간 채움 곡선

Hilbert 곡선은 정규화(normalization)된 속성을 기준으로 주소를 생성하기 때문에, 데이타의 모든 속성은 범위가 정해져 있는 스칼라 값이어야 한다. 속성 정규화를 위해 본 논문에서는 모든 센서가 속성들의 범위를 알고 있으며, 모든 센서가 동일한 정규화 기법과 동일한 Hilbert 곡선 생성방법을 가지고 있다고 가정한다. 본 논문에서는 다음과 같은 단순 정규화 기법을 사용한다. 정규화된 속성은 모두 0~1 사이의 실수 값을 가지게 된다.

$$a_N = \frac{a - a_{\min}}{a_{\max} - a_{\min}}$$

a : 센서에서 측정된 속성 값

a_N : 정규화된 데이타의 속성 값

a_{min} : 속성 값이 가질 수 있는 최소 값

a_{max} : 속성 값이 가질 수 있는 최대 값

그림 5는 선형화된 센서 네트워크에 Hilbert 곡선으로 분할된 다차원 데이타 공간을 할당하는 모습이다. (Hilbert 곡선을 이용한 데이타 공간 선형화 방법은[6,8]을 참조하라.) 초기에 센서는 선형화된 순서대로 동일한

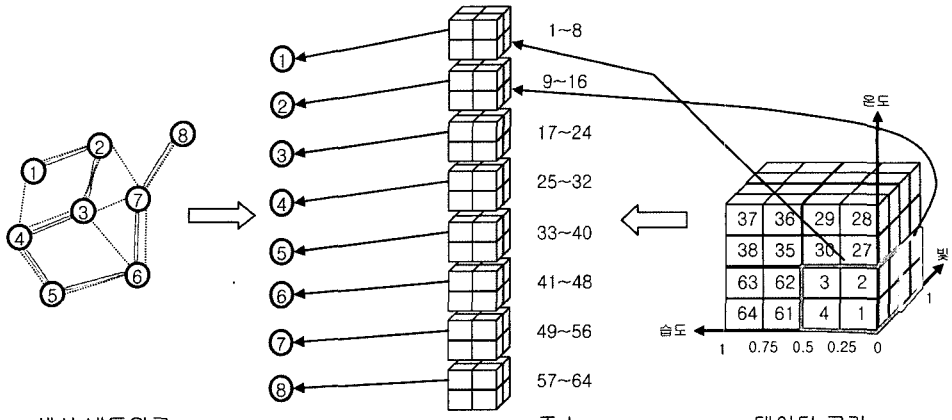


그림 5 Hilbert 곡선으로 지정된 주소의 센서 할당

크기의 주소 영역을 할당 받는다. Hilbert 곡선에 의해 생성되는 주소의 지역성 특성을 센서 네트워크에서도 유지하기 위해서 이웃 센서(직접 통신이 가능한 센서)에 인접한 주소를 할당한다. 예를 들어 그림 5에서 격자 31, 32, 33, 44는 인접한 데이타 영역들이기 때문에 지역적으로 이웃한 센서 4, 5에 할당됨으로써 31~34를 포함하는 질의가 요청됐을 때 적은 개수의 부분 질의를 사용하여 질의를 처리할 수 있다. Hilbert 곡선의 특성으로 인해 인접한 주소에 해당하는 데이타 영역들은 비교적 인접한 데이타 영역들이기 때문에, 인접한 주소가 이웃 센서들이 아닌 멀리 떨어진 센서에 할당되면 영역 질의를 수행할 때 더 많은 부분 질의가 필요하고, 멀리 떨어진 여러 센서로 질의를 보내야 한다.

4. 동적 데이타 분산

센서는 계산 능력이 낮고 저장 공간이 협소한 값싼 기기이다. 하지만, 배터리의 교환이나 재충전이 거의 불가능하고, 넓은 영역에 많은 수의 센서가 배포되기 때문에 비용을 고려하지 않을 수 없다. 이러한 점에서, 센서 네트워크의 동작시간을 연장하는 것은 매우 중요하며, 에너지가 고갈된 센서의 역할을 다른 센서에게 이양하면 네트워크의 동작시간을 연장할 수 있다. 또한, 각 센서가 가진 저장공간은 제한되어 있기 때문에, 특정 데이타 영역의 데이타가 집중적으로 생성될 경우 그 데이타 영역을 담당하는 센서의 저장 공간이 부족해 질 수 있다. 본 논문에서는 초기 시작 시에는 센서 네트워크가 균등하게 데이타 영역을 할당받도록 한 후, 운영과정에서 데이타가 일부 센서로 편중될 때 데이타를 재분산함으로써 저장공간 부족 문제도 해결할 수 있다.

4.1 과부하와 데이타 영역 조정

분산할 데이타 양을 결정하기 위해서는 센서에 가해지는 부하량(load)을 정량적으로 계산하여야 한다. 센서 전력 소모의 대부분은 무선 통신에 소모되므로, 질의 결과로 반환되는 데이타의 양에 의해 많은 영향을 받는다. 즉, 센서의 부하는 센서가 가지고 있는 데이타의 양과 그 데이타 영역에 수행되는 질의 빈도(query frequency)에 비례한다고 볼 수 있다.

정의 1. 센서에 가해지는 부하(load)는 그 센서가 저장하고 있는 데이타의 양과 그 데이타 영역에 수행되는 질의의 곱으로 정의한다.

$$L_q = \sum_k e_k q_k$$

L_q : 센서 q에 가해지는 부하의 양
e_k : 데이타 주소 k에 저장된 데이타
q_k : 데이타 주소 k의 질의 빈도

□

'이웃 센서'는 다른 센서를 통하지 않고 직접 무선 통신이 가능한 센서이다. 즉, 센서의 무선 반경 내에 존재하며 장애물에 의해 전파 방해를 받지 않는 센서들을 의미한다. 그리고 '인접 센서'는 주소 영역이 인접한 센서를 지칭한다. 인접 센서는 보통 이웃 센서 중에서 선택되며, 주소 영역을 이양하게 되는 센서이다.

'과부하(overload)'는 어떤 센서가 주위의 다른 센서에 비해 많은 전력을 소모하는 상황이다. 과부하 여부는 데이타 재분산을 수행하는 기준이 되며, 본 논문에서는 다음과 같이 과부하 여부를 결정한다. 처음 센서가 배치되었을 때는 센서의 초기 전력량 혹은 저장 공간을 기준으로 그 값이 절반 이하로 줄었을 때를 과부하로 판단하고 데이타 분산을 수행한다. 이후에는 센서의 소비 전력량이나 남은 저장 공간(free storage)이 이전 과부하 시점을 기준으로 그 절반 이하로 줄었을 때를 과부하 시점으로 정의한다. 과부하된 센서는 자신의 데이타 영

역을 축소하고 인접 센서에게 데이타 영역 및 그 영역에 저장된 데이타를 이양한다.

과부하 시 데이타 영역의 이동은 과부하 센서와 인접한 두 센서간의 부하 차이에 의하여 상대적으로 결정된다. 그렇기 때문에, 센서가 과부하로 판단되었다 하더라도 지역적으로 고르게 부하가 분산되었을 경우에는 데이타 분산이 거의 이루어지지 않는다. 또한 과부하 센서보다 인접 센서의 부하량이 더 클 경우에도 데이타 분산은 이루어지지 않는다.

정의 2. 데이타 재분산 시, 이양할 데이타 공간의 크기 Δpq는 다음과 같이 계산된다. 이동되는 데이타 영역은 인접 센서와의 부하량 차이와 부하가 발생한 센서의 데이타 영역 크기에 비례한다.

$$\Delta_{pq} = \frac{p_{max} - p_{min}}{2} \times \frac{L_p - L_q}{L_p}$$

Δ_{pq} : 센서 p에서 q로 이양될 주소 영역의 크기
p_{max} : 센서 p가 담당하는 최대 주소
p_{min} : 센서 p가 담당하는 최소 주소
L_p : 센서 p의 부하량

□

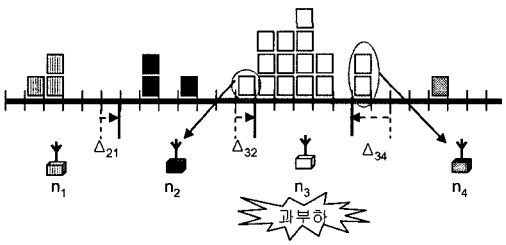


그림 6 과부하 시 데이타 영역의 조정

특정 센서에 과부하가 발생하지 않는 상황에서도 모든 센서는 시간이 지남에 따라서 스스로 과부하 상황이라고 판단하게 된다. 그런데 데이타 분산이 과부하 센서와 그 인접 센서 사이에서만 수행되면 특정 데이타 영역으로 부하가 오랜 시간 집중될 경우에는 인접센서에게만 부하가 전이될 수 있으므로 데이타 분산에 참여하는 센서는 자신의 또 다른 인접 센서에게 분산 요청을 전파하여 데이타 분산의 지역성을 극복한다. 즉, 과부하 센서로부터 일부 데이타를 이양 받은 센서는 자신의 부하를 기준으로 다시 인접 센서에게 데이타 분산 요청을 보낸다. 그림 6에서 센서 n_3에 과부하가 발생하면 데이타 분산 요청은 인접 주소 공간을 가진 센서 n_2, n_4에게 전달된다. 이후 n_2는 다시 인접 센서 n_1에게 데이타 분산 요청을 전파하여 n_2에게 더해진 부하를 줄여준다.

5. 동적 주소 기반 라우팅

데이타 기반 저장 방식의 센서 네트워크에서 라우팅은 데이타 주소와 밀접한 관계가 있다. 이는 주소가 데이타의 목적지를 의미하며, 라우팅은 데이타를 목적지로 전송하는 방법이기 때문이다. 데이타 기반 저장 방식에서 라우팅은 센서의 주소를 참조하여 전송될 센서를 선택하는데, 위치 기반 라우팅인 GPSR은 지리적 위치를 주소로 사용하므로 동적 데이타 분산을 가능하게 하려면 라우팅 테이블이 필요하다. 하지만, 전역 라우팅 테이블을 유지하게 되면 테이블 갱신에 소비되는 전력 소모가 크기 때문에, 본 논문에서와 같이 센서 데이타를 동적으로 분산하는 응용에는 적합하지 않다. 본 절에서는 이웃 센서 정보만을 이용하여 동적으로 변화하는 주소를 반영할 수 있는 라우팅 기법을 제안한다.

5.1 라우팅 구조의 구성

GPSR이 위치 정보를 그리디 라우팅한다면 본 라우팅은 주소 값을 그리디 라우팅한다. 즉, 목적 주소가 현재 라우팅을 수행하는 센서의 주소보다 큰 값을 가질 경우, 이웃 센서 중 가장 큰 주소 값을 갖는 센서로 질의를 전송하고, 그렇지 않을 경우 인접 센서 중 가장 작은 주소를 갖는 센서로 라우팅한다. 이 방법은 이웃한 센서에게 항상 인접 주소가 할당되는 경우에는 무난히 목적 센서를 찾아갈 수 있지만, 센서 네트워크는 특성상 센서들이 무작위적으로 배치되기 때문에 네트워크의 선형화 과정은 이웃하지 않는 센서를 인접 센서로 선택할 수도 있다. 이렇게 되면 주소 영역이 연속하지 않게 할당될 수 있기 때문에 그리디 라우팅(greedy routing)만으로는 확실한 데이타 전송을 보장할 수 없다. 예를 들면, 그림 7에서는 인접한 주소를 가지는 센서 6과 7이 서로 통신할 수 없기 때문에, 센서 4에서 그리디 라우팅으로 주소 70을 가진 데이타를 전송할 때, 센서 6으로 전송되어 교착 상태에 빠지게 된다. 교착 상태를 피하기 위해 센서 선형화 과정에서 다음의 두 가지 추가 정보를 사용한다.

- T_{max} : 주소가 증가하는 방향으로 인접 노드를 탐색할 때 접근 가능한 최대 주소
- T_{min} : 주소가 감소하는 방향으로 인접 노드를 탐색할 때 접근 가능한 최소 주소

T_{min}와 T_{max}는 네트워크 선형화 과정에서 함께 설정되는데, 이 값들은 현재 경로를 통해 도달할 수 있는 주소 범위를 알려주기 때문에, 효율적이고 정확한 라우팅을 가능하게 한다.이제 센서 4에서 시작된 주소 70에 관한 질의는 센서 3을 통해 목적 센서인 센서 7에 도착할 수 있다. 라우팅 알고리즘은 5.2절에서 자세히 설명된다.

센서 선형화 과정은 다음 센서를 선택하는 순행 과정

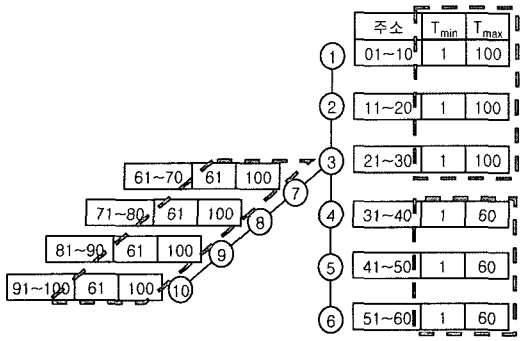


그림 7 두 개의 가지를 가지는 센서 네트워크의 정보 설정

과, 순서가 할당되지 않은 센서를 찾기 위해 순행 과정을 뒤돌아가는 역행 과정으로 나뉘는데, T_{min}은 순행 과정에서, T_{max}는 역행 과정에서 설정된다. 예를 들어, 그림 8에서 10개의 센서에게 1~100까지의 주소를 할당한다고 하자. 시작점 센서 1은 자신에게 주소 1~10을 할당하고 최소 주소인 1을 T_{min} 값으로 설정한다. 그리고 센서 2를 선택하여 선형화 과정을 계속한다. 센서 2는 11~20의 주소 영역을 자신에게 할당하고 이전 센서 1과 동일하게 T_{min} 값을 1로 설정한다. 이러한 방식으로 순행 과정에서 모든 센서들은 이전 센서와 동일한 T_{min} 값을 가지기 때문에, 센서 1~6은 T_{min} 값 1을 가진다(그림 8(a)). 센서 6에 도착하면 선택 가능한 이웃 센서가 존재하지 않기 때문에 지금까지 할당된 최대 주소값 60으로 T_{max} 값을 설정하며 역행 과정을 시작한다. 결국 센서 6에서 센서 3까지 역행하는 과정에서 센서들은 T_{max}=60을 설정하게 된다(그림 8(b)). 다시 순행 과정이 시작되어 센서 7~10은 T_{min}=61을 설정하고, 역행 과정에서 센서 7~10, 1~3에 T_{max}=100을 설정한 후 종료된다(그림 8(c)와 8(d)). 센서 네트워크가 구성 완료되면 경로에 따라 접근 가능한 데이타 주소 범위가 그림과 같이 설정된다.

5.2 라우팅 알고리즘

교착 상태를 해결하기 위하여 라우팅은 두 가지 모드를 거친다. 그 하나는 경로 찾기 모드로서, T_{min}, T_{max} 값을 참조하여 목적 주소를 포함하는 경로를 찾는다. 경로가 발견되면 이후에는 그리디 라우팅만으로 목적 주소를 포함한 센서에 도달함을 보장할 수 있다. 그리디 라우팅은 단순히 목적 주소에 가장 가까운 주소를 가진 이웃 센서로 데이타를 전송한다. 라우팅할 데이타의 주소를 D라고 하자. 라우팅을 시작하는 센서는 먼저 경로 찾기 모드로 라우팅을 시작한다. 이웃 센서 중 $T_{min} < D$

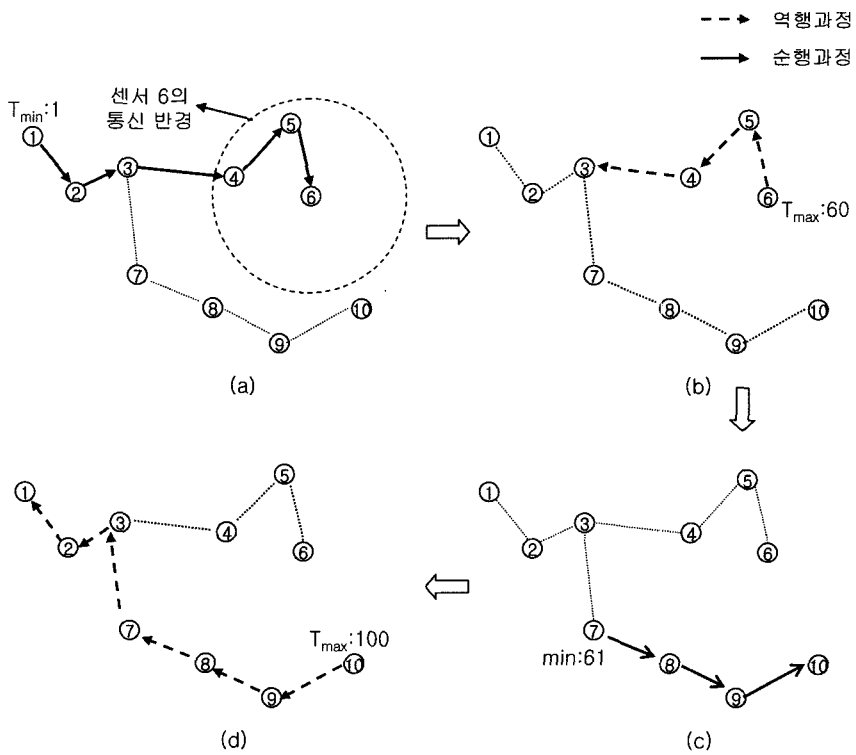


그림 8 선형화 과정에서 센서 라우팅 정보 설정

$< T_{max}$를 만족하는 (T_{min}, T_{max}) 쌍을 가지는 이웃 센서가 없을 경우, 목적 주소를 포함하고 있는 경로를 찾을 때까지 가장 작은 주소를 가진 센서로 라우팅한다(작은 주소를 가지는 센서는 항상 같거나 더 큰 T_{min}, T_{max}영역을 가지기 때문에 경로를 찾을 수 있음이 보장된다. 즉, 센서 s1의 주소 < 센서 s2의 주소 이라면 (T_{min}(s1) <=T_{min}(s2), T_{max}(s2)<=T_{max}(s1)이다). 경로 찾기 모드는 $T_{min}< D < T_{max}$인 (T_{min}, T_{max}) 쌍을 가지는 센서에 도착하였을 때 중단된다. 이후부터는 그리디 라우팅(greedy routing)이 수행된다. 그리디 라우팅은 목적 주소와 가장 가까운 주소를 갖는 이웃 센서로 라우팅한다. 라우팅은 항상 경로 찾기 모드로 시작하여 그리디 모드로 끝난다(라우팅 알고리즘의 Pseudo-Code를 부록에 기술하였다).

6. 실 험

실험은 가상의 센서 네트워크 환경을 시뮬레이션(simulation)하여 수행하였다. 시뮬레이션 환경은 다음과 같다.

- 센서 네트워크의 크기: 800m × 800m
- 센서의 무선 반경 (radio radius): 100m
- 센서의 밀도: 14(센서 하나 당 평균 14개의 이웃 센서)

6.1, 6.2절의 실험에서 센서는 장애물이 존재하지 않는 네트워크 영역에 무작위(random)로 배치되었다. 즉, 센서는 자신의 무선 반경 내에 해당하는 거리에 있는 모든 센서와 통신이 가능하다. 데이타는 5개의 속성으로 이루어져 있으며, 네트워크 생성 초기에 센서당 평균 100개의 데이타를 생성하여 센서에 저장하였다. 이후 네트워크가 동작 중단될 때까지 질의를 수행하였다. 센서의 동작 시간은 데이타 저장과 질의 결과 처리에 소모되는 전력으로 측정되었다.

6.1 데이타 공간 할당과 네트워크 동작 시간

DIM에서 데이타 영역을 분할하는 방식을 자세히 살펴보면, 두 센서가 아주 인접한 위치에 존재할 경우, 센서 밀도와는 상관 없이 한 센서는 영역의 대부분을, 다른 센서는 나머지 작은 부분만을 담당하게 됨을 알 수 있다. 이러한 문제점으로 인해, DIM은 센서간의 데이타 영역 크기의 편차가 매우 크다. 실험적으로 최대 데이타 영역을 가지는 센서는 평균보다 5.5배 정도 더 큰 데이타 영역을 할당 받는다. 이와 같은 이유로, DIM에서 센서 전력 소모는 데이타와 질의가 고르게 분포되어 있는 환경에서조차 센서간 전력 소모 편차가 매우 커 동작을 줄이는 원인이 된다. 그림 9는 질의가 모든 데이타 영역에 고르게 분산되어 있을 때 DIM 방식과 제안하는 방식의 센서 전력 소모 차이를 나타낸 것이다. 질의는 다차원 영역의 5%, 10%, 20%를 포함하는 크기로 데이타 영역 전체에 고르게 수행하였다. 측정 시점은 DIM의 센서 중 센서 하나가 동작 중단되는 시점에서 모든 센서가 동일한 데이타 영역을 가지는 경우에는 센서들이 비교적 균일하게 전력을 소모하지만 DIM은 센서들이 처리하는 데이타 양의 편차가 크기 때문에 몇몇 센서들이 대부분의 전력을 소모할 동안 나머지 센서들은 거의 사용되지 않음을 볼 수 있다.

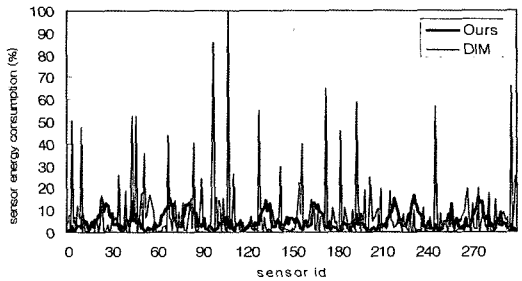


그림 9 질의가 고르게 분포되는 환경에서 센서의 전력 소모

6.2 불균일 데이타와 동적 데이타 분산

이 실험에서는 데이타의 각 속성이 독립적으로 정규분포(normal distribution)를 따르며 평균과 표준편차는 각각 0.5, 0.1로 설정되었다. 질의 역시 각 속성의 질의 영역 중심이 평균 0.5, 표준편차 0.1을 가지고, 범위는 평균 0, 표준 편차 0.1을 가지는 편중된 상황을 만들었다. 그림 10은 10%의 센서가 동작 중단된 시점에서의 DIM과 우리 시스템의 전력 소모를 각 센서의 전력 소모량에 따라 정렬하여 나타낸 결과이다. DIM은 10%의 센서가 모든 전력을 소모할 때까지 60% 이상의 센서들이 10% 미만의 전력을 소모하였다. 하지만, 동적 데이타 분산을 사용한 우리 시스템은 10%의 센서가 동작 중단되는 시점에서 네트워크 내의 거의 모든 센서들이 60% 이상의 전력을 사용하는 것을 알 수 있다. 소모된 센서 전력의 총합이 크다는 것은 그만큼 많은 많는 노드들이 질의를 처리하기 위해 동작했음을 시사한다.

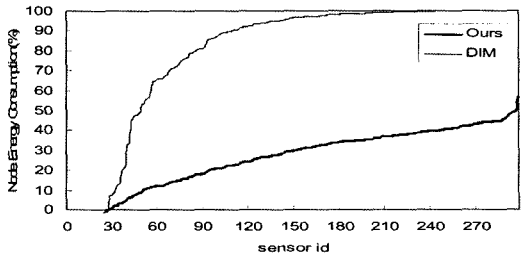


그림 10 불균일 데이타와 편중된 질의 상황에서 센서의 전력 소모

다음으로는 실제 네트워크의 동작 시간을 비교한 결과이다. 하나의 센서가 동작 중단될 때까지의 네트워크 동작 시간은 본 논문에서 제안하는 방법이 DIM에 비해 15배 정도 우수하였다. 그림 에서 볼 수 있듯이 DIM은 센서의 부하 분산에 매우 취약하기 때문에 단 하나의 센서 동작 중단도 허용치 않는 mission-critical application에는 매우 부적합하다. 전체 센서의 15%가 동작 중단(node failure)되는 시점에서, 우리의 방법은 DIM에 비해 150%정도 오래 동작함을 알 수 있다. 부하가 분산되면, 모든 센서가 하나의 시점에서 대부분 비슷한 양의 전력 소모를 보이기 때문에, 그림 8과 같이 대부분의 센서가 비슷한 동작 시간을 가지는 것을 알 수 있다.

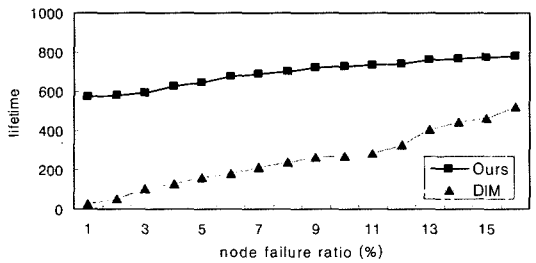


그림 11 센서 동작 중단에 따른 네트워크 동작시간 비교

6.3 라우팅 성능 비교

DIM은 센서의 위치 정보를 기반으로 GPSR 라우팅 기법을 사용한다. GPSR은 장애물을 만나기 전에 미리 우회할 수 있는 방법이 없기 때문에, 장애물을 만났을 경우 퍼리미터 라우팅(perimeter routing)을 사용하여 장애물을 우회한다. 최악의 경우 GPSR는 네트워크의 가장자리를 따라 아주 긴 경로로 데이타를 라우팅하기도 한다. 하지만 우리의 센서 네트워크 구조는 위치가 아닌 연결 정보를 기반으로 하기 때문에, 장애물의 존재와 무관하게 비슷한 성능을 보이게 된다(그림 12). 일반적으로, 장애물이 존재할 경우 지리 정보를 사용하는 라우팅이 위치 정보 기반 라우팅(GPSR 등) 보다 더 효율적이다[5].

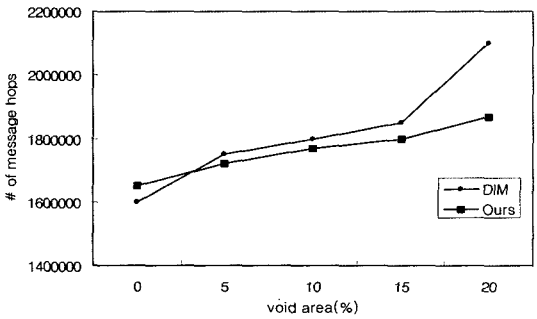


그림 12 void 영역 변화에 따른 라우팅 성능 비교

7. 결론 및 향후 과제

데이타 기반 저장 방식의 센서 네트워크에서는 데이타 분산이 동작 시간에 큰 영향을 미친다. 해시 함수(hash function)를 사용하면 비슷한 데이타가 네트워크 전역에 분산되기 때문에 부하 분산에 유리하지만 영역 질의를 수행하기에는 적합하지 않다. 반대로, 영역 질의는 데이타가 클러스터링(clustering) 되어야 효과적으로 수행될 수 있기 때문에, 데이타 분산의 목적과 상충한다. 현재까지 제안된 다차원 영역 질의 시스템은 데이타 분산을 전혀 고려하고 있지 않아, 센서의 배치 상황에 따라서 네트워크 동작 시간의 편차가 심하다.

본 논문은 다차원 영역 질의를 효율적으로 지원하면서 편중된 데이타와 질의 환경 하에서도 데이타 분산을 통해 네트워크의 동작 시간을 연장할 수 있는 방안을 제안하였다. 효율적인 영역 질의를 위해 다차원 데이타 영역을 Hilbert 공간 채움 곡선으로 분할하여 선형화된 센서 주소공간에 고르게 할당하였다. 데이타 공간 할당 시 이웃한 센서가 인접한 데이타 영역을 가질 수 있도록 하여, 효율적인 영역 질의와 데이타 분산을 가능하게 한다. 과부하된 센서는 동적으로 자신의 데이타 영역을 축소하고 관련 데이타를 이웃 센서에게 전달한다. 데이타 분산 요청은 데이타 분산이 국지적으로만 이루어지는 것을 방지하기 위하여 인접 센서로 계속 전파되어 많은 수의 센서 자원을 데이타가 집중된 데이타 영역에 배치한다.

본 논문은 센서 간의 데이타 공유를 허용하지 않았다. 데이타 영역을 배타적으로 사용하게 되면, 하나의 데이타 영역에 대부분의 데이타가 집중되는 극단적인 경우에는 데이타 분산이 불가능하다. 그러므로 센서들이 데이타 영역을 공유할 수 있는 방안에 대한 추후 연구가 필요하다. 또한, 동적 데이타 분산으로 데이타를 이웃 센서들로 이양할 경우, 데이타를 이양 받은 이웃 센서 역시 과부화 되어 원래 센서로 다시 데이타를 이양하는 현상이 반복될 수 있다. 이는 질의 처리에 필요한 에너지 사용을 줄이는 목적과 달리 데이타 분산 과정에서 필요 이상의 에너지 사용을 초래하는 문제를 가지고 올 수 있어, 이를 효과적으로 해결하는 방법에 대한 연구도 필요하다.

참 고 문 헌

[1] S. Ratnasamy, D. Estrin, R. Govindan, B. Karp, L. Yin S. Shenker, and F. Yu. *Data-centric storage in sensornets.* In ACM First Workshop on Hot Topics in Networks, 2001.

[2] Xin Li, Young Jin Kim, Ramesh Govindan, and Wei Hong, *Multi-dimensional Range Queries in*

Sensor Networks, Proceedings of the 1st international conference on Embedded networked sensor systems, ACM Press, pp. 63~75, 2003.
[3] B. Karp and H. Kung. *Greedy Perimeter Stateless Routing* In Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing, pp. 243~254, 2000.
[4] James Newsome and Dawn Song. *GEM: Graph EMbedding for Routing and Data-Centric Storage in Sensor Networks without Geographic Information.* SenSys 2003.
[5] A. Rao, C. Papadimitriou, S. Ratnasamy, S. Shenker, and I. Stoica. *Geographic Routing without Location Information.* In MobiCom, 2003.
[6] B. Moon, H. V. Jagadish, C. Faloutsos, and J. Saltz. *Analysis of the clustering properties of Hilbert space-filling curve.* IEEE Trans. on Knowledge and Data Engineering, pp 124~141, 1996.
[7] JK Lawder. *Using State Diagrams for Hilbert Curve Mappings.* Technical Report JL2/00, Birkbeck College, University of London, 2000.
[8] H. V. Jagadish, *Linear clustering of objects with multiple attributes*, International Conference on Management of Data, Proceedings of the ACM SIGMOD 1990.

부 록

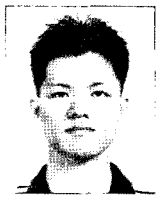
다음은 라우팅 알고리즘의 Pseudo-Code이다. 각 노드를 기술하는 구조체 'node'에 4개의 주소값과 노드의 레벨 값을 가지고 있고, 그 구조는 다음과 같다.

```
struct node {
    min_addr;  // 노드가 가진 최소 주소 값
    max_addr;  // 노드가 가진 최대 주소 값
    twig_min;  // 주소가 단순 감소하는 방향으로 라우팅했을 때
               // 도달할 수 있는 최소 주소 값
    twig_max;  // 주소가 단순 증가하는 방향으로 라우팅했을 때
               // 도달할 수 있는 최대 주소 값
    level;     // 노드의 레벨 값; 네트워크 초기 설정시 값이 정해짐
}
```

각 노드에서 라우팅 요청을 받았을 때 수행되는 작업은 다음과 같다. 각 노드에서 다음 대상 노드를 입력으로 *route*함수를 재귀적으로 호출하게 된다.

```
// INPUT: n은 라우팅을 수행하는 노드, dest_addr은 목적 주소
// OUTPUT: dest_addr을 가진 노드를 반환한다.
node route(node n, int dest_addr)
{
    node[] neighbor_set = n.getNeighbors();

    if (n.min_addr <= dest_addr <= n.max_addr)
    {
        // 목적 주소가 현재 노드의 주소 범위 내에 있을 경우
        return n;
    }
    else if (n.twig_min <= dest_addr <= n.twig_max) {
        // 목적 주소가 현재 노드가 포함된 가지 내에 있을 경우
        node next_node = neighbor_set 중 dest_addr과 가장 가까운 주소를 가진 노드
        return route(next_node);
    }
    else {
        // 현재 경로로 라우팅이 불가능 할 경우
        node next_node = neighbor_set 중 가장 작은 주소를 가진 노드;
        return route (next_node);
    }
}
```



임 용 훈

2003년 2월 연세대학교 기계전자공학부 컴퓨터과학전공 학사. 2005년 2월 한국과학기술원 전산학과 석사. 2005년 1월~현재 삼성전자 기술총괄 소프트웨어센터 근무

정 연 돈

1994년 2월 고려대학교 전산과학과 학사 1996년 2월 한국과학기술원 전산학과 석사. 2000년 8월 한국과학기술원 전자전산학과 전산학전공 박사. 2000년 9월~2001년 8월 한국과학기술원 정보전자연구소 Post-Doc. 연구원. 2001년 9월~2003년 2월 한국과학기술원 전자전산학과 전산학전공 연구교수. 2003년 3월~2006년 2월 동국대학교 컴퓨터공학과 교수. 2006년 3월~현재 고려대학교 컴퓨터학과 교수. 관심분야는 XML Database, Stream Data Processing, Mobile Databases, Sensor Networks, Database Systems 등

김 명 호

1982년 서울대학교 컴퓨터공학과 학사 1984년 서울대학교 컴퓨터공학과 석사 1989년 미국 Michigan State University 전산학과 박사. 현재, 한국과학기술원 전산학전공 교수. 관심분야는 데이타베이스 시스템, 분산시스템, XML, Sensor Networks 등