센서 네트워크에서 동적 영역 분할을 이용한 다차원 범위 질의 인덱스†

강홍구°, 김정준, 홍동숙, 한기준
건국대학교 컴퓨터정보통신공학과
{hkkang°, jjkim, dshong, kjhan}@db.konkuk.ac.kr

A Multi-dimensional Range Query Index using Dynamic Zone Split in Sensor Networks

Hong-Koo Kang°, Joung-Joon Kim, Dong-Suk Hong, Ki-Joon Han
Dept. of Computer Information & Communication Engineering, Konkuk University

요 약

최근 데이타 중심 저장 방식의 센서 네트워크에서 다차원 범위 질의를 위한 인덱스들이 제시되고 있다. 기존에 제시된 다차원 범위 질의 인덱스는 일반적으로 다차원 속성 도메인과 센서 노드의 공간 도메인을 직접 매핑하여 데이타를 관리하는 구조로 되어있다. 그러나, 이러한 구조는 센서 노드의 공간 도메인을 정적으로 분할하기 때문에 센서 노드를 포함하지 않는 영역이 생성되어 데이타 저장 및 질의 처리에서 불필요한 통신이 발생하는 문제가 있다.

본 논문은 이러한 문제를 해결하기 위해 센서 노드의 공간 도메인이 센서 노드를 포함하도록 센서 네트워크 영역을 동적으로 분할하는 다차원 범위 질의 인덱스를 제안한다. 제안하는 인덱스는 센서 노드의 위치에 따라 센서 네트워크 영역을 동적으로 분할하여 데이타 저장 및 질의 처리시 목적 영역으로의 라우팅 경로를 최적화한다. 그리고, 분할된 영역은 모두 센서 노드를 포함함으로 센서 노드에서 발행하는 저장 부하를 분산시켜 전체 네트워크에서 발생하는 전체 통신비용을 줄인다. 실험 결과 제안한 인덱스는 DIM보다 전체 센서 네트워크와 hotspot의 통신비용에서 각각 최대 35%, 60%의 성능 향상을 보였다.

1. 서 론

센서 네트워크는 다양한 데이타를 측정하는 센서 노드들로 구성되어 있다[1]. 이러한 다양한 데이타를 효율적으로 처리를 위해서는 1차원 데이타에 대한 질의보다 다차원 데이타에 대한 질의가 효율적이다. 이와 관련하여 최근 데이타 중심 저장 방식 기반 센서 네트워크에서 다차원 범위 질의를 위한 인덱스들이 제시되고 있다[2,3].

기존에 제시된 다차원 범위 질의 인덱스는 일반적으로 다차원 속성 도메인과 센서 노드의 공간 도메인을 직접 매핑하여 데이타를 관리하는 구조로 되어있다. 그러나, 이러한 구조는 센서 노드의 공간 도메인을 정적인 크기로 분할하기 때문에 센서 노드를 포함하지 않는 영역이 생성되어 데이타 저장 및 질의 처리에서 불필요한 통신이 발생하는 문제가 있다[4,5].

따라서, 본 논문에서는 이러한 문제를 해결하기 위해 센서 노드의 공간 도메인이 센서 노드를 포함하도록 센서 네트워크 영역을 동적으로 분할하는 다차원 범위 질의 인덱스인 DDIM(Dynamic and Distributed Index for Multi-dimensional data)을 제안한다. DDIM은 센서 노드의 위치에 따라 센서 네트워크 영역을 동적으로 분할하여 데이타 저장 및 질의 처리에서 데이타 전송시 목적 센서 노드로의 라우팅 경로를 최적화한다. 또한, 분할된 영역은 모두 센서 노드를 포함하도록 하여 특정 센서 노드에서 발행하는 데이타 저장 및 질의 처리의 집중을 분산시켜 전체 네트워크에서 발생하는 전체 통신비용을 줄인다. 실험 결과 제안한 인덱스는 DIM보다 전체 센서 네트워크와 hotspot의 통신비용에서 각각 최대 35%, 60%의 성능 향상을 보였다.

† 본 연구는 중소기업청의 중소기업기술혁신개발사업의 연구결과로 수행되었음.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대해 기술한다. 3장에서는 본 논문에서 제시하는 DDIM의 구조와 알고리즘을 설명하고, 4장에서는 실험을 통해 DDIM에 대한 성능을 평가하고, 5장에서 결론 및 향후 연구에 대해서 언급한다.

2. 관련 연구

2.1 GHT (Geographic Hash Table)

GHT는 데이타의 값을 기반으로 지리적 위치를 생성하여, 그 위치에 가장 가까운 센서 노드에 데이타를 저장하는 인덱스이다[2]. 그러나, GHT는 해시 테이블을 이용하기 때문에 점 질의는 효율적으로 수행할 수 있으나 범위 질의 수행은 비효율적이다. 또한, GHT는 유사한 값을 가지는 데이타를 지리적으로 멀리 떨어진 위치에 저장하기 때문에 범위 질의를 처리하기 위해서는 센서 네트워크 전역에 흩어진 여러 센서 노드들에게 부분 질의를 전송해야 하는 문제가 있다.

2.2 DIM (Distributed Index for Multi-dimensional data)

DIM은 다차원 속성 값을 기반으로 센서 네트워크의 공간 영역을 서로 매핑하면서 지리적으로 인접한 센서 노드에 데이타를 저장하는 인덱스이다[5]. DIM은 센서 네트워크 영역의 x좌표와 y좌표를 번갈아 가면서 나누어, 하나의 영역에 하나의 센서만 남게 될 때까지 영역을 분할하고, 분할된 영역에 근접한 센서에 해당 영역의 주소를 갖는 데이타를 저장한다. 그리고, 데이타 저장과 질의 처리에서 데이타 전송시 목적 센서 노드의 위치를 기반으로 하는 GPSR 라우팅 기법[2]을 사용한다.

그러나, DIM은 데이타 관리를 위해 영역을 정적으로 분할하기 때문에 센서를 포함하지 않은 데이타 영역 발생하고, 특정 노드의 데이타 저장 부하가 증가하는 문제가 있다.

3. DDIM

3.1 영역 생성

DDIM에서 각 센서 노드를 포함하는 영역은 센서 노드 위치에 따라 서로 다른 크기로 생성된다. DDIM의 영역을 생성하는 과정은 다음과 같다. 먼저, 전체 센서 네트워크를 x좌표와 y좌표를 번갈아가면서 나누어, 하나의 영역에 두 개 이하의 센서가 남게 될 때까지 반복한다. 그리고, 분할된 영역에 한 개의 센서 노드가 남게 되면 분할을 종료하고, 두 개의 센서 노드가 남게 되면 각 센서 노드를 포함하는 영역 간의 크기 차이가 최소가 되도록 축을 선택하고 분할을 수행한다.

DDIM의 각 영역은 영역 생성 과정에서 분할을 수행할 때마다 분할 축과 영역 구분자, 분할 축의 위치 코드를 나타내는 비트 스트링 형태의 영역 코드를 구성한다. 표 1은 8비트로 구성된 영역 코드를 나타낸다. 따라서, n번의 분할이 수행되면 8*n비트의 영역 코드가 생성된다.

표 1 8비트 영역 코드

분할 축 (1비트)	영역 구분자 (1비트)	위치 코드 (6비트)
X 축: 0 Y 축: 1	왼쪽 (아래쪽): 0 오른쪽 (위쪽): 1	000000~111111

그림 1은 DDIM을 구성할 센서 노드들의 예를 나타내고, 그림 2는 그림 1에서 나타내는 센서 노드들로 구성한 DDIM의 영역 코드와 경계를 나타낸다.

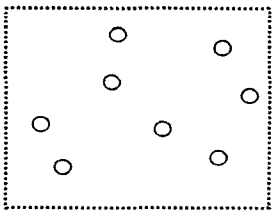

그림 1 DDIM을 구성할 센서 노드 예

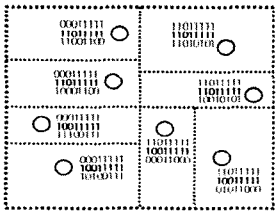

그림 2 DDIM의 영역 코드와 경계

그림 2의 각 영역들은 분할을 3번 수행하여 생성되었으므로 24비트의 영역코드를 가지게 된다.

이처럼, DDIM의 각 영역은 센서 노드 위치에 따라 분할 축의 위치가 달라지기 때문에 영역 간의 크기가 서로 다르고 각 영역은 센서 노드를 하나씩 포함하게 된다. 따라서, 센서 네트워크에 있는 각 센서 노드와 영역 간의 매핑이 쉽게 이루어진다.

그림 4는 그림 3에서 나타난 영역 코드를 트리 형태로 나타낸 것이다. 사각형으로 표현된 루트와 중간 노드는 분할 축과 분할 축의 위치코드를 가지고 있고, 원으로 표시된 리프 노드는 각 영역을 나타낸다. 그리고, 영역 코드는 루트 노드에서 리프 노드까지 경로를 나타낸다.

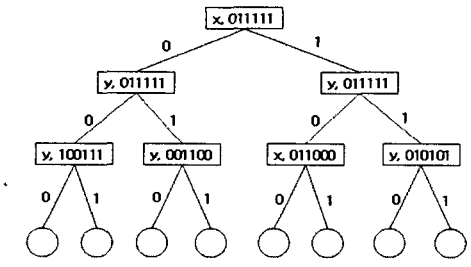

그림 3 DDIM의 영역 트리

DDIM의 각 영역에 센서 노드가 추가되거나 새롭게 발견될 경우 통신 범위 내에 있는 이웃 센서 노드와 통신하여 영역 경계를 갱신해야 한다. DDIM의 센서 노드 영역 생성 알고리즘은 그림 4와 같다.

```
Algorithm : Build_Zone(a)
1:  Begin
2:    if(CountSensor(Zone(A)) == 2) then
3:      if(SelectAxe(Zone(A)) == x-axis) then
4:        newX = (a.x + b.x)/2;
5:        SplitZone(Zone(A), newX);
6:      else
7:        newY = (a.y + b.y)/2;
8:        SplitZone(Zone(A), newY);
9:      end if
10:     UpdateZoneCode(Zone(A));
11:   end if
12: End
```

그림 4 영역 생성 알고리즘

그림 4의 영역 생성 알고리즘은 센서 노드 A를 가지는 영역에 새로운 센서 노드 a가 발견되면 두 노드 사이의 영역 경계를 갱신하도록 되어있다.

3.2 데이타 저장

DDIM에서 생성된 데이타를 적합한 센서 노드에 저장하기 위해 데이타 값을 해싱하여 영역 코드를 생성한다. 데이타 저장을 위한 영역 코드 생성은 모든 데이타 값을 0과 1사이로 일반화되어 있다는 가정하에 속성 값을 차례대로 영역 트리의 루트 노드에 있는 분할 축과 분할 축의 위치코드와 비교하여 영역 코드를 할당하는 과정을 리프 노드까지 반복 수행하여 이루어진다. 예를 들어, 저장할 데이타가 D=<0.3, 0.8>이면 영역 코드는 (00011111 11011111 11001100)이 생성된다. DDIM의 데이타 저장 알고리즘은 그림 5와 같다.

```
Algorithm : Insert_Event(e)
1:  Begin
2:    If(Equal(Code(c), Code(e)) and Is_Internal()) then
3:      Store(c, e);
4:    else
5:      B = CountNeighbor(c);
6:      while(B)
7:        Y = GetNeighbor(c);
8:        If(Closer(Y, c, Code(e)) then
9:          c = Y; B--;
10:       end if
11:     end while
12:     if(NoCloser(c)) then
13:       c = RightRouting(c, Code(e));
14:     end if
15:     Send(e);
16:   end if
17: End
```

그림 5 데이타 저장 알고리즘

그림 5의 데이타 저장 알고리즘은 데이타 d를 해싱해서 생성된 영역 코드를 이용하여 목적 영역의 중심 위치를 찾아 데이타를 전송하고, 데이타가 목적 영역에 도달하면 영역에 포함된 센서에 데이타를 저장하도록 되어있다.

3.3 질의 처리

DDIM은 임의의 센서 노드에서 요청되는 질의를 관련된 영역

으로 분해하고 해당 영역으로 질의를 전달한다. DDIM의 질의 분해는 질의 조건으로 주어진 속성 값을 해싱하여 생성된 영역 코드를 영역 트리의 노드에 있는 분할 축과 분할 축의 위치코드와 비교해 겹치는 질의 범위를 분해한다. 이 분해 과정은 루트 노드부터 리프 노드까지 반복되어 수행된다. 예를 들어, 영역 코드가 (00011111 11011111 11001100)인 센서 노드 A는 질의 Q=<0.3-0.8, 0.6-0.9>를 Q1=<0.5-0.8, 0,6-0.9>과 Q2=<0.3-0.5, 0.6-0.9>로 분해한다. DDIM의 질의 처리 알고리즘은 그림 6과 같다.

```
Algorithm : Query_Process(q)
1:  Begin
2:    if(Request(q)) then
3:      if(Split(Code(q)) then
4:        SendQuery(NextNode(q), q);
5:      end if
6:    else
7:      if(Is_Internal(Resolve(q)) then
8:        Append(q);
9:      else
10:       SendResponse(r, Code(q));
11:     end if
12:   end if
13: End
```

그림 6 질의 처리 알고리즘

그림 6의 질의 처리 알고리즘은 질의 q를 분해하여 해당 영역에 요청하고 질의 요청과 반대로 질의 결과를 응답하도록 되어있다.

4. 성능 평가

DDIM의 성능 평가를 위하여 C#으로 DIM과 DDIM을 구현하고, 저장 및 질의 처리에서 발생하는 전체 네트워크에서의 통신비용과 hotspot에서의 통신비용을 서로 비교 평가하였다. 성능 평가를 위한 환경은 표 2와 같다.

표 2 성능 평가 환경

조건	설정값
센서 노드 통신 범위	70m
센서 네트워크 크기	100m x 100m ~ 600m x 600m
센서 노드 개수	100개 ~ 600개

전체 네트워크의 통신비용은 각 센서 노드가 1,000개의 데이타를 생성하는 동안 100번의 다차원 범위 질의 수행을 수행하여 측정된 통신 횟수로 평가하였다. 전체 네트워크에서 발생한 통신비용은 그림 7과 같다.

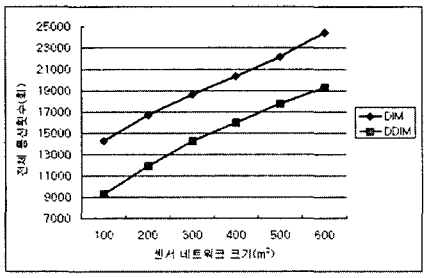

그림 7 전체 네트워크 통신비용

그림 7과 같이 센서 네트워크 크기가 커짐에도 전체 통신비용이 최대 35%가 향상되었다. 이는 DDIM이 데이타 저장 및 질의 처리에서 데이타 전송시 목적 센서 노드로의 라우팅 경로를 줄였기 때문이다.

전체 네트워크 통신비용을 평가한 것과 같은 조건에서 실험하여 얻어진 hotspot 통신비용은 그림 8과 같다.

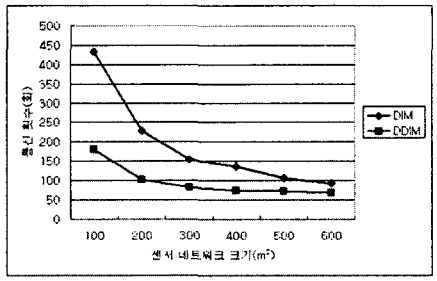

그림 8 hotspot 통신 횟수

그림 8과 같이 DDIM이 DIM보다 hotspot 통신비용이 최대 60%가 향상되었다. 이는 DDIM이 센서 노드를 포함하지 않는 영역 생성을 하지 않음으로써 특정 노드에 통신이 집중되는 것을 분산시켰기 때문이다.

5. 결론 및 향후 연구

본 논문에서는 데이타 중심 저장 방식의 센서 네트워크에서 제시된 기존의 다차원 범위 질의 인덱스의 문제를 해결하기 위해 센서의 공간 도메인을 센서들이 균등하게 가지도록 동적 분할을 이용한 다차원 범위 질의 인덱스를 제안하였다. 제안한 인덱스는 데이타 저장 및 질의 처리에서 데이타 전송시 목적 센서 노드로의 라우팅 경로를 최적화하고, 특정 센서 노드에 집중되는 부하를 분산시켜 전체 네트워크에서 발생하는 전체 통신비용을 효율적으로 줄일 수 있다.

향후 연구로는 데이타 중심 저장 방식의 센서 네트워크에서 센서 노드가 비대칭적으로 분포된 환경에 대한 성능 평가와 보다 hotspot 통신비용을 줄이기 위한 알고리즘 개선이 있다.

참고문헌

[1] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin, "Data-Centric Storage in Sensornets," In Proc. ACM SIGCOMM Workshop on Hot Topics in Networks, 2002.

[2] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "GHT: A Geographic Hash Table for Data-Centric Storage," In Proc. of the First ACM International Workshop on Wireless Sensor Networks and Applications, 2002.

[3] D. Ganesan, D. Estrin, and J. Heidemann, "DIMENSIONS: Why Do We Need a New Data Handling Architecture for Sensor Networks?," In Proc. of the First Workshop on Hot Topics in Networks, 2002.

[4] B. Greenstein, D. Estrin, R. Govindan, S. Ratnasamy, and S. Shenker, "DIFS: A Distributed Index for Features in Sensor Networks," In Proc. of 1st IEEE International Workshop on Sensor Network Protocols and Applications, 2003.

[5] X. Li, Y. J. Kim, R. Govindan, and W. Hong, "Multi-dimensional Range Queries in Sensor Networks," In Proc. of Sensys, 2003.