센서 네트워크에서 이벤트 검출 및 필터링을 위한 경로기반 네트워크-내 조인 프로세싱 방법

(Path-based In-network Join Processing for Event Detection and Filtering in Sensor Networks)

전 주 혁 † 유 재 수 †† 김 명 호 †††
(Juhyuk Jeon) (Jae Soo Yoo) (Myoung Ho Kim)

요 약 센서 네트워크의 다양한 응용 분야들 중에서 널리 사용되고 있는 것이 바로 이벤트 검출(event detection)이다. 이벤트 검출 작업은 사용자가 미리 정한 조건테이블과 센서 노드들로부터 수집된 데이타들 간 조인연산을 사용하여 쉽고 편리하게 수행될 수 있다. 또한 이벤트 검출을 위해 조인연산을 사용할 경우 네트워크-내 조인연산을 수행함으로써 통신 비용을 줄일 수 있다. 본 논문에서는 에너지 효율적으로 네트워크-내 조인연산을 수행하는 알고리즘인 PBA를 제안한다. PBA는 각 노드에서 베이스스테이션으로 데이타를 전송하는 경로 상에서 조건테이블을 나누어 저장하여 네트워크-내 조인연산을 수행한다. PBA는 각 노드들이 자신의 레벨값을 보고 저장할 부분을 식별할 수 있기 때문에 기존의 방법보다 조건테이블 전송 비용을 훨씬 더 적게 발생시킨다. 뿐만 아니라, 기존의 방법이 네트워크의 밀도에 비해 조건테이블의 크기가 클 경우 제 성능을 발휘하지 못하는 반면, 본 논문에서 제안하는 방법인 PBA는 그러한 제약 없이 대부분의 경우에도 효율적으로 동작한다. 실험 결과는 PBA가 대개의 경우에 효율적으로 동작하며, 특히 조건테이블의 크기가 네트워크 밀도에 비해 비교적 크거나 네트워크의 라우팅 트리의 높이가 큰 경우에는 기존의 방법에 비해 상당한 비용 절감 효과가 있다는 것을 보여준다.

키워드 : 센서 네트워크, 이벤트 검출, 필터링, 경로기반, 조인 프로세싱

Abstract Event-detection is an important application of sensor networks. Join operations can facilitate event-detection with a condition table predefined by a user. When join operations are used for event-detection, it is desirable, if possible, to do in-network join processing to reduce communication costs. In this paper, we propose an energy-efficient in-network join algorithm, called PBA. In PBA, each partition of a condition table is stored along the path from each node to the base station, and then in-network joins are performed on the path. Since each node can identify the parts to store in its storage by its level, PBA reduces the cost of disseminating a condition table considerably. Moreover, while the existing method does not work well when the ratio of the size of the condition table to the density of the network is a little bit large, our proposed method PBA does not have such a restriction and works efficiently in most cases. The results of experiments show that PBA is efficient usually and especially provides significant cost reduction over existing one when a condition table is relatively large in comparison with the density of the network, or the routing tree of the network is high.

Key words : Sensor Networks, Event-detection, Filtering, Path-based, Join Processing

· 본 연구는 한국과학재단 특정기초연구(R01-2006-000-10809-0)의 지원으로 수행되었음
† 학생회원 : 한국과학기술원 전산학과 연구원
jhjeon@dbserver.kaist.ac.kr
†† 종신회원 : 충북대학교 전기전자컴퓨터공학과 교수
yjs@chungbuk.ac.kr
††† 종신회원 : 한국과학기술원 전산학과 교수
mhkim@dbserver.kaist.ac.kr
논문접수 : 2006년 8월 3일
심사완료 : 2006년 10월 30일

1. 서 론

무선 네트워킹 기술과 초고밀도 집적 회로 생산기술의 발달은 컴퓨팅 능력은 물론 통신 및 센싱(sensing) 기능까지 고루 갖춘 소형의 센서 노드들로 구성된 센서 네트워크를 실현 가능하게 하였다[1]. 하나의 센서 네트워크는 여러 개의 센서 노드들로 구성되며, 조류 서식지 관찰[2], 건물의 결함 식별[3] 등 여러 분야에 사용 된

다. 하지만 센서 네트워크를 사용하는 어떠한 분야든지 결국 데이타를 수집하여 처리하는 것을 목표로 한다는 점에서 맥락을 같이한다.

[4]에서 제안한 TinyDB는 사용자가 질의를 통해 센서 네트워크에서 데이타를 수집하거나 수집한 데이타를 처리할 수 있게 하는 일종의 질의 처리기이다. TinyDB에서는 센서 네트워크를 통해 수집된 데이타들이 SENSORS라는 하나의 관계형(relational) 테이블에 속하게 된다. 또한 TinyDB는 사용자가 여러 분야에서 일관된 방식으로 센서 네트워크를 이용할 수 있도록 다양한 종류의 질의를 지원하여 사용자의 편의를 도모한다. 하지만 [4]에서 소개된 TinyDB는 SENSORS 테이블과 센서 네트워크 밖에서 작성된 정적(static) 테이블 간의 조인(join) 연산을 제공하지 않았다.

이벤트 검출(event detection)을 목적으로 센서 네트워크를 사용할 경우, SENSORS테이블과 센서 네트워크 밖에서 만들어진 정적 테이블 간의 조인연산이 유용하게 쓰일 수 있다[5]. 센서 네트워크의 여러 응용 분야들 중에서 폭넓게 사용되고 있는 것이 바로 이벤트 검출이다. 이벤트란 특정 센서 노드에서 수집된 데이타가 사용자가 미리 제시한 조건들을 만족시키는 상황을 가리킨다. 이벤트 검출 과정에서 사용자가 요구하는 조건의 수가 상당히 많을 수 있기 때문에 이러한 조건들을 하나의 테이블에 저장해두고 이 테이블과 SENSORS 테이블 간의 조인연산을 수행하여 이벤트를 검출하는데 이용한다면 매우 유용할 것이다. 여기서 사용자가 요구하는 조건들을 저장한 테이블을 조건테이블(condition table)이라 부르기로 한다. 앞서 언급한 정적 테이블은 바로 이러한 조건테이블에 해당한다.

이벤트 검출을 위해 조인연산을 이용할 경우, 네트워크 내에서 조인연산을 수행함으로써 에너지 소비를 크게 줄일 수 있다. 대개의 이벤트 검출은 공장 내 기기의 이상이나 공정 상의 에러 식별[6], 혹은 석유화학 공정 상에서 위험 물질 저장소의 안전 규정 준수[7] 등과 같은 응용 분야에 집중해 있기 때문에 조건을 만족하는 튜플이 전체 수집된 튜플에 비해 매우 적을 것이라고 예상할 수 있다. 이는 조인연산의 선택도(selectivity)가 낮다는 것을 의미한다. 선택도가 낮은 조인연산이 네트워크 내에서 수행되면, 불필요한(조인되지 않을) 데이타들이 베이스스테이션(base station)까지 전송되기 전에 미리 필터링(filtering)할 수 있기 때문에, 데이타 전송에 드는 에너지 소비를 줄일 수 있다. 대신 각 센서 노드들에 부착된 프로세서들이 조인연산을 수행하는데 필요한 에너지가 추가로 소비된다. 하지만 전력 소비 관점에서 1 비트를 전송하는 것이 800개의 인스트럭션(instruction)을 실행하는 것에 해당하기 때문에[8], 통신 비용을 줄이는 것이 연산 비용을 줄이는 것보다 전체 효율 면에서 더 중요하다. 만약 조인연산을 센서 네트워크 밖에서 수행한다면, 질의를 한번 수행할 때마다 모든 센서 노드에서 수집된 데이타들이 멀티-홉(multi-hop) 경로를 따라 베이스스테이션으로 전송되어야 한다. 이는 상당한 양의 통신 비용을 유발시키므로 에너지 제한적인 센서 네트워크 환경에 적합하지 않다. 따라서 [4,8,9]에서의 네트워크-내 프로세싱(in-network processing)과 같이 네트워크-내 조인연산을 수행하는 것이 더 에너지 효율적이다.

조인연산을 센서 네트워크 안에서 수행하려면 우선 조건테이블을 센서 네트워크 상에 저장하여야 한다. [5]에서 제안한 REED(Robust Efficient filtering and Event Detection in sensor networks)는 몇 개의 노드들로 이루어진 그룹(group)마다 조건테이블을 하나씩 저장한 후에, 그룹단위로 네트워크-내 조인연산을 수행하는 방법이다. 여기서의 그룹이란 저장 공간의 합이 조건테이블을 저장하기에 충분한 노드들의 집합을 뜻한다. REED는 그룹 형성 알고리즘(group formation algorithm)을 통해서 그룹들을 형성하는데, 이 과정에서 조인연산을 수행할 때 필요한 데이타 전송 횟수를 최소화하기 위해 그룹 내 노드들이 서로 방송 범위(broadcast range) 내에 있도록 제한한다. 또한 각 그룹의 노드들마다 저장해야 할 조건테이블의 부분(horizontal fragment)을 베이스스테이션에서 결정하고 관련 정보를 저장한다. 따라서 최악의 경우 각 노드마다 형성된 그룹 수만큼의 조건테이블을 전송해야 하므로 조건테이블 전송 비용이 클 뿐만 아니라, 네트워크 밀도(density)[1]가 충분히 높지 않으면 그룹 수가 감소하여 성능이 떨어진다는 단점이 있다.

본 논문에서는 이와 같은 REED의 문제점을 해결하기 위해, 일정한 길이의 경로를 따라서 네트워크-내 조인연산을 수행하는 방법인 경로기반 네트워크-내 조인연산 알고리즘(Path-Based in-network join Algorithm: PBA)을 제안한다. PBA는 로컬(local) 정보만을 이용하여 고정된 길이의 경로 상에 조건테이블을 나누어 저장한 다음, 수집된 데이타들이 베이스스테이션으로 전송되는 과정에서 경로를 따라 저장된 조건테이블과 조인되게 하는 방법이다. PBA의 경우 모든 노드들이 자신이 저장할 조건테이블의 부분을 식별할 수가 있기 때문에 각 노드마다 조건테이블을 한번만 전송하면 된다. 따라서 조건테이블 전송 비용이 REED보다 훨씬 적다. 또한 그룹 형성 과정 없이 라우팅 트리(routing tree)만을 참조하여 조건테이블을 저장하기 때문에, 네트워크 밀도와

1) 각 노드의 방송 범위 안에 존재하는 평균 센서 노드들의 수로 정의된다.

상관없이 네트워크-내 조인연산을 효율적으로 수행할 수 있다. 수행된 실험 결과는 PBA의 이러한 특징을 잘 보여준다.

본 논문의 구성은 다음과 같다. 2장에서는 센서 네트워크의 개요와 함께 네트워크-내 조인연산에 대한 설명 및 REED의 동작 과정을 설명한다. 3장에서는 PBA의 소개 및 알고리즘의 세부사항을 설명한다. 4장에서는 실험을 통해서 각 알고리즘을 비교하고, 통신 비용 그래프를 통한 가시적 성능 분석을 한다. 마지막으로 5장에서는 본 논문의 연구 내용을 정리 및 요약한다.

2. 배경 지식 및 관련 연구

2.1 센서 네트워크 개요

사용자가 센서 네트워크를 이용하여 데이타를 수집하는 과정은 다음과 같다. 사용자가 베이스스테이션에 필요한 데이타를 얻기 위한 질의를 입력하면, 베이스스테이션은 해당 질의를 네트워크-내 각 노드들에게 전송한다. 질의를 받은 노드들은 질의를 보고 수집한 데이타들을 베이스스테이션으로 보낼지 여부를 판단한다. 질의에 의해 선택된 데이타를 수집한 노드들은 해당 데이타를 베이스스테이션으로 전송한다. 이 때, 대부분의 경우 한 센서 노드의 통신반경이 베이스스테이션과의 거리에 비해 매우 작기 때문에 베이스스테이션이 센서 네트워크-내 다른 노드들에게 데이타 요청(보통 질의의 형태인) 메시지를 보내거나 각 노드에서 베이스스테이션으로 수집된 데이타를 전송할 때 멀티-홉 경로를 거칠 수 밖에 없다. 따라서 베이스스테이션과 각 센서 노드 간 통신을 위한 경로를 제공하는 라우팅 알고리즘(routing algorithm)이 필요하다.

본 논문에서는 현재까지 제안된 여러 라우팅 알고리즘들 중 비교적 간단한 트리 기반 라우팅 알고리즘(tree-based routing algorithm)[5,8]을 적용하였다. 트리 기반 라우팅 알고리즘은 다음과 같이 동작한다. 우선, 루트 노드(root node)[2]가 라우팅 트리 구성 메시지를 방송한다. 이 메시지에는 메시지를 보내는 노드의 ID와 레벨(level)이 포함되며, 루트 노드의 경우 레벨은 0이 된다. 트리 구성 메시지를 받은 노드들 중에서 아직 레벨이 정해지지 않은 노드들은 트리 구성 메시지를 자신에게 전송한 노드들 중 하나를 자신의 부모로 정하게 되며, 부모의 레벨에 1을 더한 값을 자신의 레벨로 정한다. 이 후에 각 노드들은 부모 노드를 통해서 루트 노드로 수집한 데이타를 전송하게 된다.

앞으로 이미 위에서 설명한 방식대로 라우팅 트리가 구성되어있다고 가정하도록 한다. 또한 라우팅 트리가 구성되는 과정에서 각 노드의 자식 노드들 및 이웃 노드들이 리스트 형태로 각 노드 내에 저장된다고 가정한다.

2.2 네트워크-내 조인 연산

이벤트 검출을 위해 네트워크-내 조인 연산을 이용하는 과정을 예제를 통해서 살펴보도록 한다. 그림 1은 조건테이블과 SENSORS 테이블 간의 조인연산을 이용하여 이벤트를 검출하는 예에 해당한다. 그림 1의 (a)는 각 노드 별로 수집된 데이타의 압력(Pres)과 습도(Humidity)를 수집된 시간과 함께 저장한 SENSORS 테이블을 나타낸다. 그림 1의 (b)는 센서가 부착된 곳에서 준수되어야 할 압력의 상한(Pres_thresh)과 습도의 상한(Humid_thresh)을 각 시간대 별로 포함하고 있는 조건테이블(condition table)을 나타낸다. 그림 1의 (c)는 (b)의 조건테이블을 이용하여 이벤트 검출을 하기 위한 질의의 예에 해당한다. 사용자는 그림 1의 (c)와 같은 질의를 통해서, 어떤 센서 노드에서 수집된 데이타가 조건테이블에 명시된 상한을 넘을 경우, 노드의 ID와 만족된 조건의 넘버를 1초 간격으로 얻을 수 있다. 즉, 사용자가 두 테이블(alert_table, SENSORS) 간의

Node_ID	Time	Pres	Humidity
1	1am	1.2atm	30%
2	5am	0.9atm	10%
3	2pm	2.6atm	70%
...	...	...	...

(a) 수집된 데이타 테이블: SENSORS

Condition_no	Time	Pres_thresh	Humid_thresh
1	9am	> 1.5atm	> 70%
2	10am	> 2atm	> 65%
3	11am	> 2.5atm	> 55%
...	...	...	...

(b) 조건테이블: alert_table

```
SELECT s.nodeid, a.condition_no
FROM sensors AS s, alert_table AS a
WHERE s.time = a.time
AND s.pres > a.pres_thresh
AND s.humidity > a.humid_thresh
SAMPLE PERIOD 1s
```

(c) 위의 조건테이블을 사용하는 질의

그림 1 조건테이블을 이용한 이벤트 검출 방법

2) 베이스스테이션에 직접 연결된 노드를 뜻한다. 문맥 상 별 문제가 없을 경우 베이스스테이션이라는 용어와 루트 노드라는 용어를 번갈아 사용하였다.

조인연산을 포함하는 질의를 통해서, 센서가 있는 곳의 압력이나 습도가 주어진 상한을 넘는 이벤트를 검출 할 수 있다. 만약 조건테이블 없이 기존의 질의([4]의 TinyDB에서 제공하는)만을 이용하여 위와 같은 이벤트 검출을 할 경우, 모든 시간대 별 조건들을 질의 안에 포함시켜야 하므로 질의가 상당히 길어질 수 있다. 이 때, 질의 자체가 너무 길어 한 노드에서 저장할 수 없는 경우에는 질의를 나누어 저장 및 처리해야 하는 번거로움이 따른다. 또한 시간뿐만 아니라 위치에 따른 조건을 포함할 경우 질의가 더욱 복잡해지기 때문에 질의를 작성하는 데에 꽤 많은 시간을 소비하게 될 것이다. 반면에 조건테이블을 이용한다면 조건들이 테이블 형태로 따로 저장되므로 질의에서 이를 일일이 명시할 필요가 없으며, 위의 예제와 같은 방식으로 시간 및 위치에 따른 이벤트 검출 작업을 단 몇 줄의 질의로 표현할 수가 있다. 따라서 조건들을 테이블 형태로 저장한 후 조인연산을 이용하여 질의를 처리하는 것이 훨씬 더 유용하다는 것을 알 수 있다.

2.3 REED

REED는 미리 정해진 조건테이블과 센서 노드들로부터 수집된 데이타들의 집합인 SENSORS 테이블 간의 네트워크-내 조인연산을 통해 이벤트 검출을 용이하게 하기 위한 방법이다. 이벤트 검출을 위해 조건테이블을 이용하는 네트워크-내 조인연산은 조건테이블의 크기에 따라서 크게 세 가지로 나눌 수 있다.

(i) 조건테이블을 한 노드에 저장할 수 있는 경우 - 단순 조인(Simple Join)

(ii) 조건테이블을 하나의 그룹[3] 내에 저장할 수 있는 경우 - 분산 조인(Distributed Join)

(iii) 조건테이블을 하나의 그룹 내에서도 저장할 수 없는 경우 - 부분 조인(Partial Join)

선택도가 낮은 조인연산은 수집된 데이타의 필터(filter)와 같은 기능을 하기 때문에 가능한 한 데이타가 수집된 곳으로부터 가까운 지점에서 조인연산을 수행한다면 통신 비용을 더욱 줄일 수 있을 것이다. 따라서 (i)의 경우에는 모든 센서 노드들마다 조건테이블을 저장하여 수집된 데이타와 조건테이블 간의 조인연산을 노드 안에서 수행하도록 하는 단순 조인 연산을 수행한다. (iii)의 경우에는 하나의 그룹 내에서도 조건테이블을 저장할 수 없기 때문에 [5]에서 제안한 몇 가지 전 후처리 방법들을 사용하여 부분 조인을 수행한다. 단순 조인은 간단히 적용할 수 있으며, 부분 조인은 분산 조인과는 별도로(orthogonally) 적용될 수 있다. 따라서 네트워크-내 조인연산 중 분산 조인이 핵심이 된다고 할 수 있으며, 실제로 [5]에서 제안한 REED는 이러한 분산 조인 방법에 중점을 둔 방법이다. 본 논문에서는 단순 조인 및 부분 조인은 고려하지 않으며, 분산 조인 방법인 REED만을 고려하기로 한다. 또한 본 논문에서 제안하는 방법인 PBA도 이러한 분산 조인 방법에 해당한다.

REED의 경우 하나의 조건테이블을 저장하는 단위는 노드들의 집합인 그룹이 된다. 그룹 단위의 네트워크-내 조인연산 알고리즘은 그룹 형성 과정과 연산 과정으로 나누어진다. 이 때, 조인연산을 포함하는 질의와 조건테이블의 크기 정보를 담은 메시지가 라우팅 트리를 통해서 베이스스테이션에서 각 노드로 이미 전송된 상태라고 가정한다.

2.3.1 그룹 형성 과정

만약 임의의 한 노드가 자신에게 조건테이블을 저장할 만한 충분한 공간이 없다고 판단하면, 그 노드는 즉시 그룹 형성 알고리즘을 개시한다. 이 때, 다음과 같은 두 가지 조건이 충족되어야 한다.

① 한 그룹 내 모든 노드 쌍들은 서로 방송 범위 내에 존재해야 한다.

② 그룹 내 노드들의 여유저장공간의 합은 적어도 조건테이블의 크기보다 같거나 커야 한다.

①은 네트워크-내 조인연산 과정에서 수집된 데이타를 그룹 내 모든 멤버들에게 전송할 때 드는 비용을 최소화하기 위함이며, ②는 그래야만 그룹 안에서 완전한 조인연산을 수행할 수 있기 때문이다. 자세한 그룹 형성 과정은 [10]을 참조하기 바란다. 여기서는 그룹 형성 과정의 개요만을 제시하도록 한다.

REED에서의 그룹 형성 과정은 다음과 같이 진행된다. 임의의 한 노드가 그룹 형성 공고 메시지를 방송 범위 안에 있는 노드들에게 전송하는 것으로 그룹 형성 과정이 시작 된다. 이 노드를 마스터(master) 노드라고 한다[4]. 마스터 노드로부터 메시지를 받은 노드들은 자신의 이웃 노드 리스트와 여유저장공간 크기를 담은 메시지를 마스터 노드에게 보낸다. 마스터 노드는 별도로 그룹 멤버 집합(group member set: G)과 임시 이웃 리스트(temporary neighbor list: N)를 가지며, G와 N을 각각 자신을 포함하는 단원소 집합 및 자신의 이웃 노드 리스트로 초기화한 다음, 이웃 노드들로부터 메시지를 받을 때마다 N과 메시지에 포함된 이웃 노드 리스트 간의 교집합을 취한다. 이 때, 교집합을 취한 결과가 G를 포함할 경우 해당 이웃 노드를 G에 포함시키고, N에 교집합 한 결과를 할당한다. 이와 같은 과정을 받은 모

3) 2.3.1절에서 설명할 그룹 형성 과정을 통해서 형성되는 노드들의 집합으로 정의된다.

4) 마스터 노드를 비롯한 모든 그룹 내 노드들에 대해 조인연산을 수행하는 과정이 거의 같기 때문에, 어떤 노드가 마스터 노드가 되느냐는 별로 중요하지 않다.

든 메시지들에 대해서 반복한다. 이 때, G는 항상 조건 ①을 만족한다. 앞의 과정을 반복하는 동안 G가 조건 ②를 만족하면 G에 포함된 노드들로 구성된 그룹이 형성된다. 이 후, 형성된 그룹의 각 노드들이 베이스스테이션에게 조건테이블을 요청하면, 베이스스테이션이 조건테이블을 가로로 나누어(horizontally fragmentizing) 해당 그룹의 멤버들에게 전송한다.

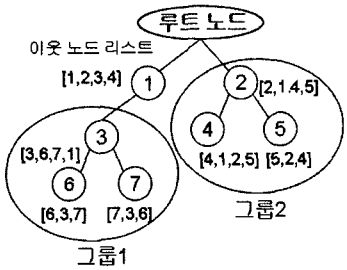
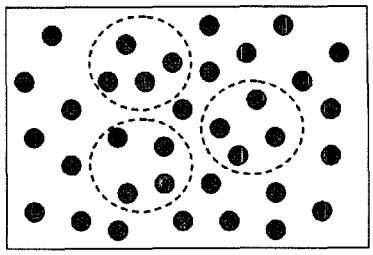


그림 2 그룹 형성 결과

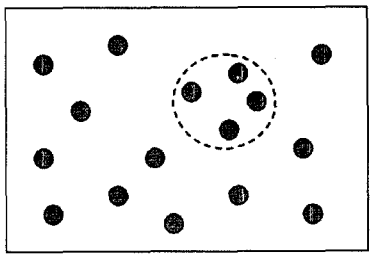
앞선 과정을 통해 형성된 그룹의 예는 그림 2와 같다. 여기서 조건테이블의 크기가 세 개의 노드에 나누어 저장할 수 있을 정도라고 가정한다. 형성된 그룹은 총 두 개이며, 하나는 노드3, 6, 7이 포함된 그룹(그룹1), 다른 하나는 노드2, 4, 5를 포함하는 그룹(그룹2)에 해당한다. 그룹1의 경우, 노드3, 6, 7의 이웃 노드 리스트들의 교집합({3, 6, 7})이 그룹에 속한 노드들의 ID를 모두 포함하고 있고 크기가 3이므로 조건테이블을 저장하기에 충분하다는 것을 알 수 있다. 그룹2의 경우에는 노드2, 4, 5의 이웃 노드 리스트들의 교집합이 {2, 4, 5}이므로, 그룹1과 마찬가지로 교집합이 그룹 멤버를 모두 포함하며 포함하고 있는 노드의 수도 조건테이블을 저장하기에 충분하다는 것을 알 수 있다. 한편, 노드1, 2, 4의 이웃 노드 리스트들의 교집합이 {1, 2, 4}이므로 노드1, 2, 4로 구성된 그룹이 형성될 수도 있었으나, 이미 노드2와 4가 그룹2에 속하기 때문에 노드1을 포함하는 그룹이 형성되지 않았다.

REED에서 주어진 조건테이블 크기에 맞춰 형성된 그룹 수는 네트워크 밀도에 비례한다. 즉, 네트워크 밀도가 낮으면 그룹이 거의 형성되지 않는다. 이는 동일한 네트워크 내에서 이벤트 검출에 사용될 조건테이블의 크기가 증가할 경우도 마찬가지다. 조건테이블의 크기가 증가한다는 것은 하나의 그룹을 형성하기 위해서 더욱 많은 노드들이 필요하다는 것을 의미하며, 네트워크 내에서 서로 방송 범위 안에 있는 노드들의 수에는 한계가 있기 때문에 결과적으로 그룹 형성의 성공률이 떨어지게 된다.

그림 3은 네트워크의 밀도가 그룹 형성 과정에 미치



(a) 네트워크 밀도가 높은 경우



(b) 네트워크 밀도가 낮은 경우

그림 3 네트워크 밀도에 따른 그룹 수 비교

는 영향을 나타낸다. 그림 3의 (a)는 네트워크의 밀도가 높은 경우를, 그림 3의 (b)는 네트워크 밀도가 낮은 경우를 나타낸다. 각 그림 상에서 회색의 작은 원은 센서 노드를 나타내며, 점선으로 된 큰 원은 직경이 센서 노드의 방송 범위에 해당하는 것으로 해당 부분에서 그룹 형성이 가능한지 여부를 판단하기 위한 것이다. 만약 점선의 원에 포함되는 노드 수가 조건테이블을 나누어 저장하기에 충분할 정도라면, 점선의 원이 포함하고 있는 부분에서 그룹이 형성될 것이라 기대할 수 있다. 그림 3의 (a)와 (b)를 비교하면, 동일한 크기의 조건테이블에 대해서 (a)의 경우가 더 많은 그룹이 형성될 수 있다는 것을 확인할 수 있다. 가령 조건테이블을 4개의 노드에 나누어 저장할 수 있다고 하면, 4개의 노드를 포함하면서 서로 겹치지 않는 점선의 원의 최대 개수는 (b)의 경우보다 (a)의 경우가 더 많다.

2.3.2 연산 과정

그룹이 형성되어 조건테이블 정보가 분배된 이 후부터 연산 과정이 시작된다. 연산 과정은 노드가 그룹에 속하는지의 여부에 따라 달라진다. 우선, 그룹에 속한 노드인 경우 자신이 수집한 데이타를 그룹의 다른 멤버들에게 방송한다. 이 후, 조인연산은 그룹 내 모든 노드들에 의해 수행된다. 그룹에 속한 노드가 아닐 때에는 다음과 같이 두 가지 경우에 따라 연산 과정이 달라진다.

① 데이타를 루트 노드로 전송하는 경로 상에 어떠한 그룹도 존재하지 않는 경우

② 데이타를 루트 노드로 전송하는 경로 상에 적어도

한 그룹이 존재하는 경우

①인 경우에는 수집된 데이타가 루트 노드까지 그대로 전송되므로 해당 데이타에 대해 네트워크-내 조인연산이 수행되지 않는다. 이는 REED를 적용하지 않았을 때와 같다. ②인 경우에는 처음으로 만나는 그룹 멤버를 통해서 해당 그룹 내 모든 노드들에게 데이타가 방송된다. 그 다음의 과정은 그룹에 속한 노드의 경우와 같이 진행된다. 따라서 형성된 그룹 수가 많을수록 각 노드에서 수집된 데이타가 네트워크-내 조인될 확률이 높다고 할 수 있다. 형성된 그룹 수는 네트워크 밀도 및 조건테이블의 크기에 영향을 받기 때문에, 네트워크-내 조인연산을 통해서 얻을 수 있는 통신 비용 절감 효과가 형성된 그룹 수에 의존한다는 사실은 네트워크 밀도 및 조건테이블의 크기가 알고리즘의 성능에 많은 영향을 준다는 것을 의미한다.

3. 제안하는 방법

REED의 경우 그룹 내 임의의 두 노드가 서로 방송범위 안에 있어야 한다는 제약 때문에 형성된 그룹 수는 센서 네트워크의 밀도에 영향을 받는다. 가령 조건테이블의 크기가 10개의 노드들의 여유저장공간 크기의 합과 같다면, 적어도 10개의 노드가 직경이 방송 범위인 원 안에 위치할 경우에만 그룹이 형성될 수 있다. 또한 그룹 별 테이블 분배에 관한 글로벌(global) 정보는 베이스스테이션만이 가지고 있기 때문에 그룹 내 각 노드들은 베이스스테이션으로부터 자신이 저장할 조건테이블 부분 데이타가 전송될 때까지 기다려야 하며, 지금까지 형성된 그룹들의 위치를 알지 못하므로 다른 그룹에 저장될 데이타는 무조건 자신보다 레벨이 낮은 노드들에게 전송해야 한다. 따라서 최악의 경우 각 노드마다 형성된 그룹 수배만큼의 조건테이블 데이타를 전송해야 한다.

본 논문에서 제안하는 PBA는 REED의 위와 같은 단점을 보완하기 위해 로컬 정보만을 이용하여 조건테이블을 분배할 수 있게 하는 동시에, 그룹을 형성할 필요없이 일정 길이의 경로를 따라서 조인연산이 수행될 수 있게 하는 경로기반 네트워크-내 조인 알고리즘이다. 따라서 PBA를 이용하면 네트워크 밀도와 상관없는 성능을 보장받는 동시에 조건테이블 전송 비용을 크게 줄일 수 있다. 그림 4는 네트워크 밀도가 낮을 때 PBA가 동작하는 모습을 나타낸다. 이 때, 조건테이블이 4개의 노드에 걸쳐 저장될 수 있다고 가정한다. 여기서 회색의 작은 원은 센서 노드를 나타내며, 각 노드 간의 선은 라우팅 트리 에지(edge)를 의미한다. 또한 점선으로 된 화살표는 데이타를 수집한 노드에서부터 해당 데이타의 조인연산 과정이 끝나는 노드까지의 경로를 나타낸다. 즉, 각 노드에서 수집된 데이타는 화살표를 따라 조인된다. 실제로 이와 같은 조인연산 과정이 거의 모든 노드에서 일어나지만, 편의상 몇 개의 화살표만을 표기하였다. REED의 경우 그림 3의 (b)에서와 같이 네트워크 밀도가 낮으면 그룹이 거의 형성되지 않기 때문에 각 노드들에서 수집된 데이타가 네트워크-내 조인될 확률이 낮다. 반면 PBA의 경우, 그림 4에서 알 수 있듯이 네트워크의 밀도가 낮을 때에도 거의 모든 노드들이 자신이 수집한 데이타가 일정 길이의 경로를 따라서 조인되게 할 수 있다.

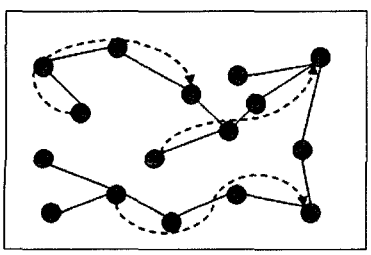


그림 4 네트워크 밀도가 낮을 때의 PBA 동작 과정

PBA는 조건테이블 분배 과정과 연산 과정으로 이루어진다. 각 과정을 설명하기에 앞서, 알고리즘을 기술하는데 필요한 가정 및 용어들을 설명하도록 한다.

3.1 가정 및 용어 설명

설명을 간단히 하기 위해서 각 노드의 여유저장공간 크기는 모두 같다고 가정한다. 또한 조건테이블의 각 튜플의 크기 역시 모두 같다고 가정한다. 따라서 앞으로 데이타(조건테이블, 노드 간 주고받는 메시지 등)의 크기를 튜플 단위로 계산하도록 한다.

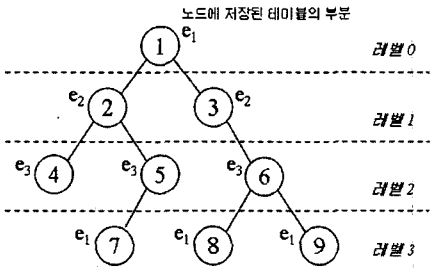
알고리즘을 기술하는데 쓰이는 용어들은 다음과 같다.

- T: 조건테이블에 속한 튜플들의 집합에 해당한다.
- t: 조건테이블의 튜플 수인 |T|를 나타낸다.
- m: 각 노드의 여유저장공간 크기를 튜플단위로 나타낸다.
- 프래그먼트(fragment): T를 m보다 작거나 같은 수의 튜플을 포함하는 $\lceil t/m \rceil$ 개의 상호 배타적인 (mutually exclusive) 부분집합들로 나눌 수 있다. 이러한 부분집합을 프래그먼트라 부르기로 한다
- FN: 주어진 조건테이블의 프래그먼트 개수에 해당한다.
- mod: 나머지 연산자이다. 가령 A mod B는 A에서 B를 나눈 나머지 값을 반환한다.

앞으로 특별한 언급이 없는 한, 위와 같은 용어들을 논문 전반에 걸쳐 사용하도록 한다. 이어지는 절들에서는 PBA를 이루는 조건테이블 분배 과정과 연산 과정을 다룬다.

	Condition_no	Time	Pres_thresh	Humid_thresh
e_1	1	9am	> 1.5atm	> 70%
	2	10am	> 2atm	> 65%
e_2	3	11am	> 2.5atm	> 55%
	4	12pm	> 3atm	> 50%
e_3	5	1pm	> 3.5atm	> 45%
	6	2pm	> 4atm	> 40%

(a) 두 튜플씩 나눠진 조건테이블



(b) 조건테이블 분배

그림 5 PBA에서의 조건테이블 분배

3.2 조건테이블 분배 과정

먼저 루트 노드에서 자식 노드들에게 조건테이블을 방송한다. 루트 노드의 자식 노드들은 자신이 저장할 조건테이블의 프래그먼트를 레벨을 통해서 식별한 후 이를 저장한다. 각 노드들이 저장할 프래그먼트의 번호는, 레벨을 L이라고 할 때 (L mod FN)+1과 같다. 이렇게 하는 이유는 다음 절에서 설명할 연산 과정에서 레벨이 FN-1이상인 어떤 노드들에 대해서도, 자신보다 레벨이 FN-1만큼 높은 조상 노드(ascendant node)에 이르기까지의 경로 상에 모든 프래그먼트들이 존재하도록 하기 위함이다. 따라서 모든 노드들은 조건테이블의 (L mod FN)+1번째 프래그먼트를 저장하며, 받은 조건테이블 데이타를 다시 자신보다 레벨이 낮은 노드들에게 전송한다. 이 과정이 네트워크 안의 모든 노드들이 프래그먼트를 저장할 때까지 반복된다. 이 때, 한 노드가 전송해야 할 데이타의 크기는 조건테이블의 크기와 같다. 즉, 각 노드들은 조건테이블을 한번씩만 전송하면 되며, 이는 형성된 그룹 수에 비례한 만큼 조건테이블을 전송해야 했던 REED보다 훨씬 통신 비용이 적게 든다는 것을 의미한다.

그림 5는 PBA의 조건테이블 분배 과정을 보여주는 예에 해당한다. 그림 5의 (a)는 조건테이블이 e_1, e_2, e_3, 이렇게 세 프래그먼트로 나눠진 모습을 나타낸다. 편의상 한 프래그먼트에 2개의 튜플이 포함된다고 가정하였다. 그림 5의 (b)는 테이블이 분배된 이 후를 나타내며 각 노드는 조건테이블의 세 프래그먼트 중 하나씩을 저장한 상태이다. 그림 5의 (b)에서 노드 1은 레벨이 0이고 0 mod 3 = 0이다. 0+1 =1이므로 e_1을 저장한다. 여기서 3는 FN에 해당한다. 마찬가지로 노드 2는 레벨이 1이고 1 mod 3 = 1이다. 1+1 = 2이므로 e_2를 저장한다. 다른 노드들도 이와 같은 과정을 통해서 자신이 저장해야 할 프래그먼트를 식별한 후 이를 저장한다. 그림 3의 (b)에서 레벨 2 이상인 노드들에 대해, 자신으로부터 레벨이 2 높은 조상 노드에 이르기까지의 경로 상에 조건테이블의 모든 프래그먼트들이 저장되어 있다는 것을 확인할 수 있다.

3.3 연산 과정

일단 조건테이블이 모든 노드들에 걸쳐서 분배되면, 수집된 데이타들과 조건테이블 간의 조인연산이 수행될 수 있는 상태가 된다. 레벨 0에서 FN-2 사이의 노드들을 제외한 모든 노드들은 자신이 수집한 데이타를 자신보다 FN-1만큼 위로 전송하여 경로 상의 각 노드들에 저장된 프래그먼트들과 수집된 데이타 간의 조인연산이 수행될 수 있도록 한다. 이 때, 자신을 포함하여 총 FN개의 노드를 거치게 되므로, FN개의 서로 다른 프래그먼트들과 수집된 데이타 간의 조인연산이 수행된다. 이는 결국 수집된 데이타와 조건테이블 간의 네트워크-내 조인연산이 빠진 부분 없이 완전하게 수행된다는 것을 의미한다. 따라서 모든 노드들에 대해 레벨이 FN-1 높은 노드에서 수집한 데이타의 조인연산이 끝나게 된다. 이 과정에서 조인된 결과들은 모두 루트 노드로 전송된다. 레벨 0에서 FN-2 사이의 노드들은 자신으로부터 루트 노드에 이르는 경로의 거리가 FN-1보다 작기 때문에 네트워크-내 조인연산을 수행할 수 없다. 이 경우 노드들은 수집한 데이타를 루트 노드로 바로 전송한다.

그림 5의 (b)에서 노드 7이 수집한 데이타가 네트워크 내에서 조인되는 상황을 고려해보자. 노드 7은 자신이 가지고 있는 프래그먼트인 e_1과 수집한 데이타 간의 조인연산을 수행한다. 이 후 노드 7은 자신이 수집한 데이타를 부모 노드인 노드 5로 전송한다. 노드 5는 노드 7로부터 받은 데이타와 노드 5에 저장된 프래그먼트 e_3에 대해 조인연산을 수행한 후 다시 노드 7의 데이타를 부모 노드인 노드 2로 보낸다. 노드 2는 노드 5와 마찬가지로 자신이 보유한 프래그먼트 e_2와 노드 7의 데이타에 대해 조인연산을 수행한다. 이 과정에서 각 노드에서 조인연산이 수행된 이 후 생성된 결과 데이타들은 즉시 자신의 부모 노드를 통해서 루트 노드로 전송된다. 노드 7에서 수집된 데이타는 앞선 과정을 통해서 조건테이블의 모든 프래그먼트들과 조인되었기 때문에 노드 2에서 조인연산이 마무리 된다. 이와 마찬가지로 레벨이 2보다 작은 노드 1,2,3을 제외한 모든 노드들에 대해, 각 노드

들이 수집한 데이타는 해당 노드로부터 레벨이 2만큼 높은 노드까지 전송되는 동안 모든 조건테이블의 프래그먼트들과 조인된다.

4. 실 험

본 논문에서는 시뮬레이션 실험을 통해 이벤트 검출 과정에서 REED와 PBA를 사용하였을 때 발생하는 통신 비용을 비교하였다. 여기서의 통신 비용이란 메시지 전송 횟수를 의미한다. 또한 네트워크-내 조인연산 알고리즘을 사용하지 않는 경우에 대해서도 통신 비용을 계산하여 다른 알고리즘들을 사용한 경우와 비교하였다. 이 경우를 NAIVE라 부르기로 한다. 이 때, 각 알고리즘들의 특성을 좀 더 명확하게 파악하기 위해, 통신 비용을 질의 실행 횟수에 상관없이 일정한 부분과 비례하여 증가하는 부분으로 나누어 비교하도록 한다. 이 때, 전자를 정적(static) 비용, 후자를 동적(dynamic) 비용이라 부르기로 한다. 정적 비용에는 라우팅 트리 구성 비용, 질의 전송 비용, 그룹 형성 비용, 조건테이블 전송 비용 등이 있으며, 동적 비용으로는 질의 실행 비용이 있다. 모든 알고리즘에 대해 동일한 라우팅 트리 구성 알고리즘을 적용하였으므로 라우팅 트리 구성 비용 및 질의 전송 비용은 모두 같다. 따라서 정적 비용으로는 그룹 형성 비용과 조건테이블 전송 비용만을 고려한다.

4.1 실험 환경

본 실험에서는 센서 노드들이 직사각형의 공간에 임의로 배치된 센서 네트워크 환경을 시뮬레이션하였다. 센서 네트워크는 주어진 크기의 가로 및 세로 길이를 가지며, 이러한 직사각형의 왼쪽 하단의 좌표값을 (0, 0)으로 하여 각 노드들의 좌표값을 임의로 정했다. 센서 네트워크의 가로길이를 R, 세로 길이를 C, 각 노드의 방송 범위를 r, 센서 네트워크의 밀도를 d라고 할 때, 센서 네트워크-내 총 노드들의 개수 N은 다음과 같은 식으로 주어진다.

$$N = \frac{R \times C}{\pi \times r^2} \times d \quad (1)$$

이 때, 길이의 단위는 미터라고 가정한다. 위의 식과 실험에 사용되는 여러 파라미터 값들을 통해서 총 노드 개수 N을 구하고, 루트 노드를 제외한 N-1개의 노드들을 센서 네트워크 내에 임의로 배치시켜서 실험하였다. 루트 노드는 센서 네트워크의 가운데에 배치시켰으며, 베이스스테이션이 루트 노드로부터 방송 범위 안에 있는 임의의 위치에 존재한다고 가정하였다. 실험에 사용되는 파라미터들과 각각의 기본값은 표 1과 같다.

이어지는 절들에서는 앞에서 설명한 실험 환경을 바탕으로 표 1의 파라미터들을 적용하여 각 알고리즘 별로 측정한 메시지 전송 횟수를 비교 분석한다. 이 때,

표 1 실험에 사용되는 파라미터 값

파라미터	기본값
네트워크 밀도(d)	10
가로×세로(R×C)	1000×1000
방송 범위(r)	30
프래그먼트 수(FN)	10
조인연산의 선택도(s)	0.1

노드들이 임의로 배치된 상황을 고려하기 때문에, 표 1의 파라미터 값들을 같게 한다고 해도 실험을 반복할 때마다 측정값이 달라질 수 있다. 즉, 센서 노드들의 배치에 따라서 각 알고리즘 별 성능 차이가 발생할 수 있다. 이러한 변화를 반영하기 위해서 모든 실험의 최종 측정값을 동일한 파라미터 값들에 대해 5번씩 실험한 결과값들의 평균으로 정했다.

4.2 실험 결과 및 분석

통신 비용을 정적 비용과 동적 비용으로 나누어 비교하도록 한다. 우선 4.2.1절에서는 정적 비용에 속하는 그룹 형성 비용 및 조건테이블 전송 비용을 비교한다. 4.2.2절에서는 동적 비용에 속하는 질의 실행 비용을 비교한다. 각 알고리즘의 정적 비용은 센서 노드의 여유저장공간 크기를 일정한 값으로 두고, 조건테이블의 크기를 증가시켜가며 측정하였다. 동적 비용은 네트워크의 가로 길이 및 밀도가 작은 경우, 보통인 경우, 큰 경우 각각에 대해서 프래그먼트 수의 네트워크 밀도에 대한 비율을 달리하여 측정하였다. 프래그먼트 수의 네트워크 밀도에 대한 비율을 변화시켜가면서 실험한 이유는 4.2.2절에서 자세히 설명하도록 한다.

4.2.1 정적 비용

표 1의 파라미터들을 적용하여 각 알고리즘 별 정적 비용을 실험을 통해서 비교하였다. 단, 조건테이블의 크기에 따라서 정적 비용이 어떻게 변하는지를 알아보기 위해 조건테이블 크기를 달리하여 실험하였다. 편의상 한 노드가 1개의 조건테이블 튜플을 저장할 수 있다고 가정하고 조건테이블이 0에서 18개의 튜플을 포함할 경우의 정적 비용을 계산하였다. 각 알고리즘 별로 발생한 정적 비용은 그림 6과 같다. NAIVE의 경우 정적 비용이 따로 들지 않기 때문에 그림에서 제외되었다. 이 때, 가로축은 조건테이블의 크기를 나타내고, 세로축은 전송된 메시지 수를 나타낸다.

그림 6을 통해서 REED가 PBA에 비해 정적 비용을 훨씬 더 많이 발생시킨다는 것을 확인할 수가 있다. 조건테이블 크기가 3인 지점을 기점으로 REED의 정적 비용이 감소하는 이유는, 조건테이블의 크기가 노드의 여유저장공간 크기에 비해 점점 더 커짐에 따라서 그룹 형성 과정을 통해 형성되는 그룹의 수가 꾸준히 감소하기 때문이다. 실제로 조건테이블의 크기가 12일 때에는

그룹이 거의 형성되지 않아 정적 비용이 0에 가까워진다는 것을 알 수가 있다.

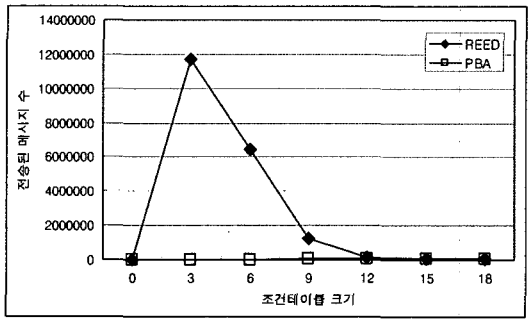


그림 6 조건테이블 크기에 따른 정적 비용

4.2.2 동적 비용

앞선 논의와 같이 REED와 PBA의 성능은 프래그먼트 수와 네트워크 밀도에 의존한다. 그림 3에서 알 수 있듯이, REED는 네트워크 밀도가 낮거나 프래그먼트 수가 많아서 그룹이 잘 형성되지 않을 경우에는 제 성능을 발휘하지 못한다. 반면 PBA의 경우에는 네트워크 밀도가 높거나 프래그먼트 수가 적을 때에는 REED에 비해서 상대적으로 더 많은 통신 비용을 발생시킨다. 이는 REED에서는 한 노드에서 수집된 데이타가 그룹 내에서 한번의 방송을 통해서 조인될 수 있는 반면, PBA에서는 프래그먼트 수-1만큼의 데이타 전송이 필요하기 때문이다. 즉, 프래그먼트 수의 네트워크 밀도에 대한 비율이 작을 때에는 REED가 PBA보다 더 효율적이며, 클 때에는 PBA가 더 성능이 좋을 것이라고 예상할 수 있다. 따라서 동적 비용을 측정할 경우에는 다음과 같이 프래그먼트 수의 네트워크 밀도에 대한 비율로 정의 되는 α 값을 달리하여 실험하였다.

$$\alpha = \frac{FN}{d}$$ (FN: 프래그먼트 수, d: 네트워크 밀도)

α 값은 두 알고리즘의 동적 비용을 비교하기 위한 좋은 척도가 된다. 이와 비슷하게 α 의 역수인 $1/\alpha$ 을 척도로 동적 비용을 비교할 수도 있다. 하지만 α 의 분모에 해당하는 네트워크 밀도는 주어진 센서 네트워크의 고유한 값이므로 이벤트 검출을 위한 질의가 바뀌거나 조인에 참여할 조건테이블이 바뀐다고 해도 변하지 않는 반면, α 의 분자인 프래그먼트 수는 사용자가 필요로 하는 조건테이블에 따라서 얼마든지 증가할 수가 있다. 그렇기 때문에 프래그먼트 수에 비례하는 α 가 동적 비용 비교에 더 적합하다. 따라서 α 값을 점점 증가시켜가면서 동적 비용을 측정하여 비교하였다. 뿐만 아니라, 센서 네트워크 상의 노드 배치에 따른 상황을 고려하기 위해서 네트워크의 가로 길이 및 밀도가 작은 경우, 보통인 경우, 큰 경우 이렇게 세 가지 경우에 대해서 α 값에 따른 동적 비용을 측정하였다.

우선, 4.2.2.1절에서는 각 알고리즘 별로 네트워크의 밀도가 작은 경우, 보통인 경우, 큰 경우로 나누어 α 값에 따른 동적 비용을 측정하여 비교하도록 한다. 4.2.2.2절에서는 네트워크의 가로 길이가 짧은 경우, 보통인 경우, 긴 경우 각각에 대해 α 값에 따른 동적 비용을 측정한 다음, 이를 서로 비교 분석하도록 한다.

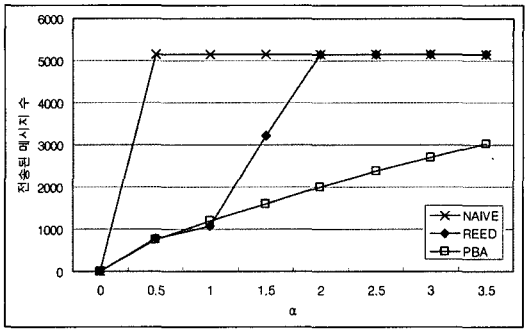
4.2.2.1 네트워크 밀도 실험

네트워크의 밀도와 각 알고리즘 간의 관계를 알아보기 위해 네트워크의 밀도가 4(작은 경우), 10(보통 경우), 14(큰 경우)인 각 경우에 대해서 α 값에 따른 동적 비용을 측정하였다. 그림 7은 네트워크 밀도에 따른 각 알고리즘 별 동적 비용을 나타낸다. 이 때, 네트워크의 밀도가 아닌 다른 모든 파라미터들을 표 1의 기본값으로 두고 실험하였기 때문에, 네트워크의 밀도가 증가함에 따라서 식 (1)에 의해 N값이 증가한다. 이는 네트워크 밀도에 비례하여 각 알고리즘의 통신 비용이 증가한다는 것을 의미한다. 실제로 그림 7의 (a)에서 (c)로 갈수록 전체적인 통신 비용이 증가한다는 것을 확인할 수 있다. 또한 모든 알고리즘에 대해 동일한 질의가 실행되었다고 가정하였으며, 편의상 질의 수행횟수를 1회로 정하였다.

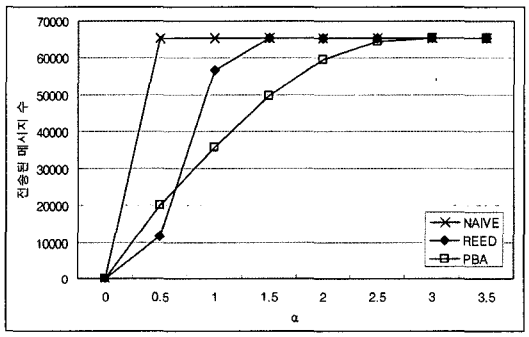
그림 7의 (a)는 네트워크 밀도가 낮을 경우를 나타내며, α 값이 1이 될 때까지 PBA와 REED가 거의 같은 양의 통신 비용을 발생시키다가 그 이 후부터는 REED의 통신 비용이 급격히 증가함을 알 수가 있다. 이는 α 값이 증가하면서 형성되는 그룹 수가 감소하기 때문이다. 그림 7의 (b)와 (c)를 보면, 그림 7의 (a)와 마찬가지로 α 값이 1이상인 경우에 PBA가 REED보다 더 성능이 낮다는 것을 확인할 수가 있다. 또한 그림 7의 (a)에서 (c)로 갈수록 PBA와 REED의 통신 비용 곡선의 기울기가 증가함을 알 수가 있다. 같은 α 값을 갖는다고 해도 밀도가 증가함에 따라서 조건테이블의 크기가 증가하기 때문에, 두 알고리즘의 통신 비용 곡선 모두 네트워크의 밀도가 낮을 경우보다 더욱 빨리 NAIVE의 통신 비용 곡선에 접근하게 된다. 따라서 네트워크의 밀도가 14보다 더 큰 경우에는 α 에 상관없이 REED와 PBA 모두 NAIVE와 비슷한 정도의 통신 비용을 발생시킬 것이라 예상할 수 있다.

4.2.2.2 네트워크 가로 길이 실험

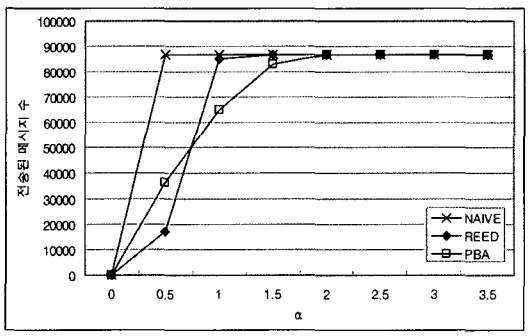
네트워크의 가로 길이가 각 알고리즘의 성능에 미치는 영향을 분석하기 위하여, 네트워크의 가로 길이가 1,000m(짧은 경우), 5,000m(중간인 경우), 10,000m(긴 경우)인 각 경우에 대해 실험하였다. 이 때, 네트워크의



(a) 네트워크 밀도 = 4

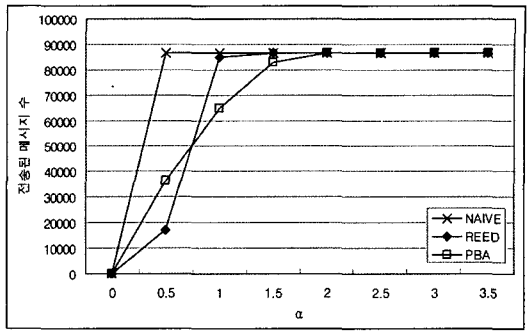


(b) 네트워크 밀도 = 10

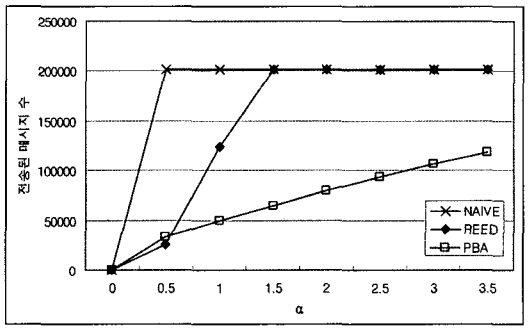


(c) 네트워크 밀도 = 14

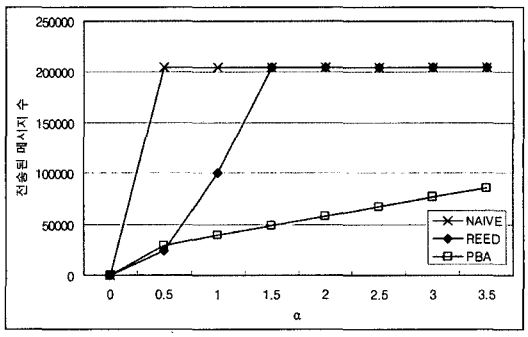
그림 7 네트워크 밀도에 따른 동적 비용



(a) 네트워크 가로 길이 = 1,000m



(b) 네트워크 가로 길이 = 5,000m



(c) 네트워크 가로 길이 = 10,000m

그림 8 네트워크의 가로 길이에 따른 동적 비용

가로 길이의 변화에 상관없이 센서 네트워크-내 노드 수인 N값을 일정하게 유지하기 위해 네트워크의 세로 길이도 함께 변하게 하였다. 즉, 네트워크의 가로 길이를 R이라고 할 때, 네트워크의 세로 길이는 1,000,000/R이 된다. 그림 8은 네트워크의 가로 길이에 따른 각 알고리즘 별 동적 비용을 나타낸다. 가로축은 α 값을, 세로축은 질의 실행 과정에서 전송된 메시지 수를 나타낸다. 이 때, 앞 절의 네트워크 밀도 실험과 마찬가지로 모든 알고리즘에 대해 동일한 질의가 실행되었다고 가정하였으며, 편의상 질의 수행횟수를 1회로 정하였다.

그림 8을 통해서 α 값이 1이상인 모든 경우에 대해 PBA가 REED보다 통신 비용이 더 적게 든다는 것을 확인할 수가 있다. 또한, 그림 8의 (a), (b), (c) 각각을 비교해보면, PBA가 REED의 성능을 추월하는 지점이 네트워크의 가로 길이가 증가함에 따라 점점 더 앞당겨짐을 알 수가 있다. 특히, 그림 8의 (c)에서 α 값이 약 0.5일 때에도 PBA가 REED만큼 효율적이라는 것을 확인할 수가 있다. 이는 네트워크의 가로 길이가 길수록 PBA가 더 효율적으로 동작한다는 것을 보여준다. PBA의 성능이 네트워크의 가로 길이에 비례하여 증가하는

이유는 바로 네트워크의 가로 길이에 비례하여 라우팅 트리의 높이도 함께 증가하기 때문이다. 라우팅 트리의 높이가 높다는 것은 레벨이 프래그먼트 수-1이상인 노드들의 개수가 증가한다는 것을 의미하며, 이는 곧 어떤 노드에서 수집된 데이타가 네트워크-내 조인될 확률이 높다는 것을 뜻한다. 또한 우리가 시뮬레이션하는 상황에서는 루트 노드가 센서 네트워크의 한 가운데에 위치해있기 때문에, 네트워크의 세로 길이를 증가시킬 경우도 마찬가지의 결과를 가져올 것이라 기대할 수 있다.

PBA와 REED 간의 통신 비용 차이가 가장 큰 경우는 그림 8의 (c)에서 알 수 있듯이, 네트워크의 가로 길이가 길면서(10,000m) α 값이 1.5일 때이다. 이 경우, PBA는 REED보다 약 4배 정도 통신 비용을 적게 발생시킨다는 것을 확인할 수 있다.

5. 결 론

본 논문에서는 이벤트 검출을 위한 효율적인 네트워크-내 조인연산 알고리즘인 PBA를 제안하였다. PBA는 각 노드가 자신의 레벨을 통해서 조건테이블의 프래그먼트를 저장하고, 각 노드에서 수집된 데이타는 일정 길이의 경로를 통해서 네트워크 내에서 조인될 수 있게 하는 경로기반 네트워크-내 조인연산 알고리즘이다. 따라서 최악의 경우 각 노드 별로 형성된 그룹 수만큼의 조건테이블을 전송해야 하는 REED와는 달리, PBA는 각 노드 당 한번의 조건테이블 전송 과정을 통해서 의도한 위치에 조건테이블을 배치시킬 수 있기 때문에 조건테이블 전송 비용이 훨씬 적게 든다. 또한, PBA는 센서 노드들이 특별히 조밀하게 배치되지 않은 상황에서도 네트워크 내에서 효율적으로 조인연산을 수행할 수 있다. 실험 결과는 이러한 PBA의 특징을 잘 보여주며, α 값이 1보다 작을 때를 제외한 대부분의 경우 PBA가 적어도 REED보다 효율적이라는 것을 보여준다. 특히, 네트워크 길이가 길 경우에는 거의 모든 α 값에 대해 PBA가 REED보다 효율적이며, 최대 REED보다 약 4배의 비용 절감 효과를 가져온다는 것을 확인하였다.

참 고 문 헌

[1] F. Zhao, and L. Guibas, Wireless Sensor Networks, pp.1-22, Elsevier, San Francisco, 2004.

[2] Mainwaring, A., Polastre, J., Szewczyk, R., Culler, D., and Anderson, J., "Wireless Sensor Networks for Habitat Monitoring," In Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, pp. 88-97, 2002.

[3] N. Xu, S. Rangwala, K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, "A Wireless Sensor Network for Structural Monitoring," SenSys, 2004.

[4] S. R. Madden, M. J. Franklin, J. M. Hellerstein and W. Hong, "TinyDB: An Acquisitional Query Processing System for Sensor Networks," ACM Trans. on Database Systems, Vol. 30, No. 1, pp.122-173, 2005.

[5] D. J. Abadi, S. R. Madden and W. Lindner, "REED: Robust, Efficient Filtering and Event Detection in Sensor Networks," In Proc. of VLDB Conf., 2005.

[6] L. Nachman, R. Kling, R. Adler, J. Huang and V. Hummel, "The Intel Mote Platform: A Bluetooth-based Sensor Network for Industrial Monitoring," In Proc. of IPSN, 2005.

[7] Stephan Haller, Taking Sensor Network Technology to A Smarter Level, IST_Results, 2006.

[8] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "TAG: A Tiny Aggregation Service for Ad-hoc Sensor Networks," In Proc. of OSDI, 2002.

[9] Yong Yao and Johannes Gehrke, "The Cougar Approach to In-network Query Processing in Sensor Networks," SIGMOD Record, vol. 31, no. 3, pp. 9-18, 2002.

[10] D. J. Abadi, et al., "REED: Robust, Efficient Filtering and Event Detection in Sensor Networks," In technical report, 2004.

전 주 혁

2005년 한국과학기술원 전자전산학과 전산학전공 학사. 2005년 3월~현재 한국과학기술원 전자전산학과 전산학전공 석사과정. 관심분야는 데이타베이스시스템, 센서 네트워크, 스트림 데이타 처리, 시멘틱 웹 등

유 재 수

정보과학회논문지 : 데이타베이스 제 33 권 제 5 호 참조

김 명 호

정보과학회논문지 : 데이타베이스 제 33 권 제 1 호 참조