센서네트워크에서 시그니처 기반 데이터 집계를 이용한 이동객체 트래킹 기법

(Tracking Moving Objects Using Signature-based Data Aggregation in Sensor Network)

김 용 기* 김 영 진** 윤 민*** 장 재 우****
(Yong Ki Kim) (Young Jin Kim) (Min Yoon) (Jae Woo Chang)

요 약 현재, 센서네트워크 기술을 이용한 많은 응용들이 개발되고 있다. 이러한 많은 응용 가운데 이동객체 트래킹 기법은 중요한 이슈 중에 하나이다. 그러나 현재 이에 대한 연구는 많은 연구가 이루어지지 않은 상태이며, 존재하는 연구는 다음과 같은 2가지 문제점을 가지고 있다. 첫째, 이동객체의 트래킹을 위해 반복적으로 센서노드를 방문해야하는 오버헤드가 발생한다. 둘째, 여러 이동객체를 동시에 지원하지 못한다. 이러한 문제를 해결하기 위해 본 논문에서는 시그니처 기반의 효율적인 데이터 집계를 이용한 이동객체 트래킹 기법(SigMO-TRK)을 제안한다. 이를 위해, 첫째, 공간 필터링 방법을 이용하여 효과적으로 이동객체들의 궤적을 집계하기 위한 지역적 라우팅 계층트리를 구성한다. 둘째, 시그니처를 사용하여 효율적으로 모든 이동객체들의 궤적에 대한 트래킹을 수행한다. 또한, SigMO-TRK를 확장하여 주어진 질의에 대한 이동객체의 유사궤적을 검색한다. 마지막으로, TOSSIM 시뮬레이터를 사용하여 제안하는 이동객체 트래킹 기법이 기존의 트래킹 기법보다 에너지 효율성 측면에서 우수함을 보인다.

키워드 : 센서네트워크, 이동객체 트래킹, 시그니처, 데이터집계

Abstract Currently, there are many applications being developed based on sensor network technology. A tracking method for moving objects in sensor network is one of the main issue of this field. There is a little research on this issue, but most of the existing work has two problems. The first problem is a communication overhead for visiting sensor nodes many times to track a moving object. The second problem is an disability for dealing with many moving objects at a time. To resolve the problems, we, in this paper, propose a signature-based tracking method using efficient data aggregation for moving objects, called SigMO-TRK. For this, we first design a local routing hierarchy tree to aggregate moving objects' trajectories efficiently by using a space filtering technique. Secondly, we do the tracking of all trajectories of moving objects by using signature in a efficient way, our approach generates signatures to method. In addition, by extending the SigMO-TRK, we can retrieve the similar trajectories of moving objects for given a query. Finally, by using the TOSSIM simulator, we show that our signature-based tracking method outperforms the existing tracking method in terms of energy efficiency.

Keywords : Sensor Network, Moving Object Tracking, Signature, Data Aggregation

1. 서 론

현재 유무선 통신 기술의 발전 및 모바일 정보기기의 보편화에 힘입어, 시간과 장소의 제약 없이 서비스를 제공할 수 있는 유비쿼터스 센서 네트워크 기술에 대한 관심이 크게 고조되고 있다[1,2,3,4,5]. 또한, 센서노드의 하드웨어 기술 발전으로 인해, 확장된 저장 공간에서 복잡한 연산을 처리하는 것이 가능하게 되었다. 이를 통해 교통,

† 본 연구는 교육과학기술부와 한국산업기술재단의 지역혁신인력양성사업으로 수행된 연구결과임.

* 전북대학교 컴퓨터공학과 박사과정, ykkim@dblab.chonbuk.ac.kr
** 삼성전자 DMC연구소 사원, glen.kim@samsung.com
*** 전북대학교 컴퓨터공학과 석사과정, miny@dblab.chonbuk.ac.kr
**** 전북대학교 전기전자컴퓨터공학부 교수, jwchang@chonbuk.ac.kr(교신저자)

논문접수 : 2009.02.16 수정일 : 1차 2009.04.27 심사완료 : 2009.04.29

시설물관리, 문화재, 환경 등의 다양한 분야에서 센서네트워크 기술을 적용한 다수의 응용서비스가 개발되고 있다.

최근, 센서 네트워크의 응용분야 중에서 이동객체의 이동경로를 추적하는 이동객체 트래킹(tracking) 기법은 중요한 연구 분야 가운데 하나이다. 센서 네트워크에서 이동객체의 트래킹에 관한 연구는 크게 이동객체의 정확한 위치를 측정하는 연구 및 센서 네트워크내의 데이터 처리 및 데이터 집합에 관한 연구이다[6,7,8,9,10,11]. 이 중에서 본 논문은 이동객체 트래킹을 위한 데이터 처리 및 데이터 집합에 관한 연구를 수행한다. 기존 이동객체 트래킹을 위한 센서 네트워크내의 데이터 처리 및 데이터 집합의 연구는 라우팅 트리를 구성하여 데이터를 수집 및 가공하여 기지국(Base Station)으로 전송하는 방법을 수행한다. 그러나, 기존 연구에서는 다음과 같은 단점을 가진다. 각 센서노드는 이동객체의 이동여부를 저장하고 있는 센서에 직접 방문을 통해 이동객체의 이동여부를 확인하기 때문에, 노드의 재방문으로 인한 오버헤드가 발생한다. 이는 제한된 에너지를 가지는 센서 네트워크에서는 많은 수의 메시지로 인해 에너지 비효율성을 초래한다. 따라서 본 논문에서는 시그니처를 이용하여 효율적으로 에너지를 관리하는 이동객체 트래킹 기법을 제안한다. 제안하는 기법은 첫째, 공간 필터링을 적용한 지역적 라우팅 계층트리를 통해 효율적으로 이동객체의 이동 정보를 집계한다. 둘째, 센서노드에서 저장하고 있는 이동객체의 이동정보를 시그니처로 구성하고, 이를 이용하여 이동객체의 트래킹을 효율적으로 처리한다. 셋째, 제안하는 이동객체 트래킹 기법의 구조를 확장하여 여러 이동객체들의 궤적 정보를 싱크노드로 수집하고, 질의 궤적과 유사한 이동객체의 궤적을 검색하는 유사궤적질의를 수행한다.

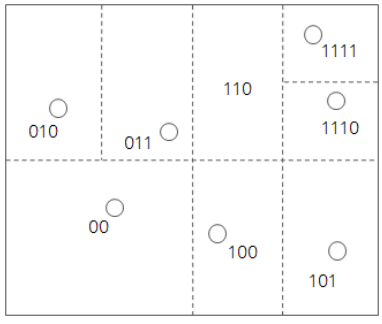
본 논문의 구성은 다음과 같다. 2장에서는 이동객체의 트래킹 기법의 관련연구를 살펴보고, 3장에서는 본 연구에서 제안한 시그니처를 이용한 이동객체 트래킹 기법을 제시한다. 4장에서는 제안하는 이동객체 트래킹 기법을 확장한 유사궤적 질의처리 기법을 제시하며, 5장에서는 이동객체 트래킹 기법 및 유사궤적 질의의 성능평가 수행한다. 6장에서는 결론 및 향후 연구 방향을 제시한다.

2. 관련 연구

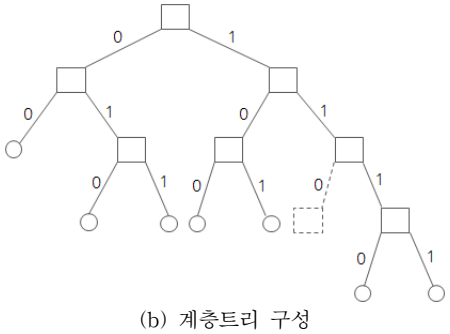
본 장에서는 센서네트워크에서 이동객체 트래킹을 위한 관련 연구를 소개한다. 센서네트워크에서 이동객체 트래킹을 위한 관련연구는 질의 영역내의 센서노드를 통해서 센싱 데이터를 집계하는 DIM[8] 및 이동객체의 움직임 빈도를 고려하여 데이터를 집계하는 STUN [10]이 존재한다.

2.1 DIM(Distributed Index for Multi-dimensional data)

DIM은 질의 영역내의 센서노드를 통해 다차원의 속성 데이터를 효율적으로 집계하기 위한 기법으로서, 해싱을 통해 분산된 계층 트리를 구성한다. 그림 1은 DIM의 전체 구조를 나타낸다. 그림 1의 (a)는 센서네트워크 분할을 나타낸다. DIM의 센서노드의 삽입에 따른 센서네트워크 분할 방법은 다음과 같다. 첫째, 센서네트워크 분할 횟수를 통해서, 분할 횟수가 홀수이면 y축을 분할한다. 분할 횟수가 짝수이면 x축을 분할한다. 둘째, y축으로 분할하는 경우, 좌측공간에는 비트 0을 할당하고 우측공간에는 비트 1을 할당한다. x축으로 분할하는 경우, 상단공간에는 비트 1을 할당하고 하단공간에는 비트 0을 할당하여 분할을 수행한다. 또한 각 영역에 할당된 비트를 통해서 그림 1의 (b)와 같이 계층트리를 구성한다. DIM은 계층트리를 이용해서 사용자로부터 입력받은 영역에 대한 질의를 수행한다. DIM의 영역질의 처리 순서는 다음과 같다. 첫째, DIM의 계층트리를 통해서 질의 영역내의 가장 상위노드를 선출한다. 둘째, 질의영역 내 최상위노드로 GPSR(Greedy Perimeter Stateless Routing) 라우팅 프로토콜[12]을 이용하여 질의를 전송한다. 마지막으로, DIM의 계층트리를 통해서 각 하위노드의 센서노드로 부터 다차원의 속성 데이터를 싱크노드로 집계한다. DIM은 해싱을 통해 구축된 라우팅 트리를 통해 지역 보존을 유지함으로써 데이터 집계를 효율적으로 처리한다. 그러나 DIM은 범위질의를 빠르게 수행할 수 있는 장점이 있는 반면, 영역 질의에 초점이 맞추어져 있기 때문에 이동객체가 움직이는 시점마다 계층트리를 재구성해야하는 단점이 존재한다.



(a) 센서네트워크 분할

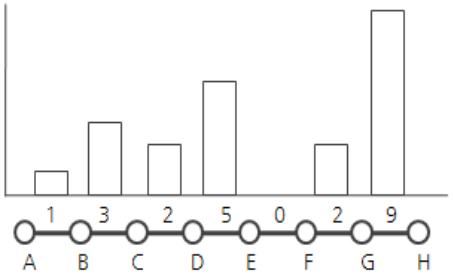


(b) 계층트리 구성

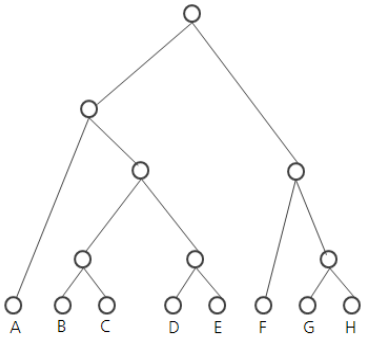
그림 1. DIM의 구조

2.2 STUN(Scalable Tracking Using Networked sensors)

STUN은 트래킹을 효율적으로 처리하기 위해 제안된 기법으로써, 이동객체의 움직임 빈도를 통해 트래킹 계층 트리(DAB)를 구축한다. STUN의 이동객체 트래킹 처리 순서는 다음과 같다. 첫째, 이동객체의 움직임 빈도 그래프를 구성한다. 둘째, 구성된 움직인 빈도 그래프를 통해 계층 트리를 구성한다. 셋째, 계층트리를 이용하여 이동객체의 이동정보를 집계한다. 마지막으로, 계층트리를 통해 센서노드를 재방문하여 이동객체의 트래킹을 수행한다. 그림 2는 STUN의 이동객체 트래킹 기법의 구조를 나타낸다. 그림 2의 (a)는 이동객체의 움직임 빈도 그래프를 나타낸다. 즉, A-H로 이루어진 센서네트워크를 이동객체가 위와 같은 빈도를 가지고 이동하였다는 것을 나타낸다. 그림 2의 (b)는 움직임 빈도를 이용해 구축한 트래킹 계층 트리를 나타낸다. STUN의 계층트리 구축 방법은 이동객체의 움직임 빈도 그래프를 통해 가장 높은 빈도를 가지는 구간부터 시작하여, 모든 구간에 대해 각 구간의 노드가 동일한 부모 노드를 가지도록 구성한다. 이는 이동객체가 동일한 구간을 이동하였을 경우, 다수의 이동정보 데이터를 집계하여 상위노드에게 전송하는 방법이다. 이를 통해 전송 메시지 수를 감소시켜 에너지 효율성을 높인다. STUN은 움직임 빈도에 따라 트래킹 계층 트리를 구성하기 때문에 움직임이 적은 이동 객체에 비해 움직임이 많은 이동 객체에 보다 효율적이다. 그러나 다수의 이동 객체를 지원하는데 있어 보다 많은 노드의 에너지를 소모하는 단점을 지니고 있다. 또한, 지속적으로 모니터링을 지원해야 하는 응용에 대해 제한점을 지니고 있다.



(a) 이동객체의 움직임 빈번도 그래프



(b) 트래킹 계층 트리

그림 2. STUN의 이동객체 트래킹 기법의 구조

3. 시그니처를 이용한 이동객체 트래킹 기법

센서네트워크에서 이동객체 트래킹 기법은 설계 시, 다음과 같은 사항을 고려해야 한다. 첫째, 자동차, 사람 등의 이동객체를 효율적으로 다루기 위하여, 지역적으로 라우팅 계층 트리를 구성하여야 한다. 둘째, 센서노드가 제한된 에너지를 지니고 있기 때문에, 데이터 처리 및 데이터 집계 기법을 통하여 송수신 데이터의 양을 줄여야 한다. 마지막으로, 데이터 모니터링을 위하여 다수의 이동객체를 지원하여야 한다.

그러나 기존의 대표적인 트래킹 기법인 DIM과 STUN은 다음과 같은 문제점을 지닌다. 첫째, 지역적 또는 이동객체의 빈도에 따라 동적으로 계층트리를 구성하므로 빈번한 재구축으로 인한 오버헤드 발생되며, 또한 n개의 센서노드에서 싱크노드로의 데이터 집계를 위해 다수(n-1개)의 중간 계층노드가 필요하여 전체적인 통신 메시지의 수가 증가한다. 이로 인해, 에너지 효율성을 위한 효율적인 라우팅 계층 트리를 구성하지 못한다. 둘째, 각 센서노드는 이동객체가 자신의 영역만을 처리하기 때문에, 이동객체가 움직이는 궤적을 트래킹하기 위해서 계층 트리의 재탐색이 필요하다. 이로 인해 송수신하는 데이터의 양이 많아진다. 마지막으로, 이동객체 트래킹을 위한 계층 트리는 한 시점 또는 이동객체의 빈도에 따라 구성하므로, 일정 범위 시간 또는 다수의 이동객체를 처리하기 위해서는 다수의 계층 트리를 요구하게 된다.

따라서 본 논문에서는 센서네트워크에서의 이동객체의 트래킹을 위한 고려사항을 반영한 새로운 시그니처 기반 [13] 데이터 집계를 이용한 이동객체 트래킹 기법(SigMO-TRK 이라 명명)을 제안한다. SigMO-TRK는 첫째, 고정된 지역적 라우팅 계층 트리를 구축함으로써, 효율적으로 이동객체를 다루며, 아울러 다수의 이동객체의 동시 트래킹 처리를 지원한다. 둘째, 기지국으로 보내는 데이터의 효율적인 집계를 위해, 이동객체의 특성을 고려하며 중복된 메시지수를 차단하기 위한 공간 필터링 기법을 적용한다. 마지막으로, 노드의 재방문을 최소화하여 메시지 양을 줄이고 효율적인 이동객체 관리를 위하여, 이동객체의 이동정보를 시그니처로 구성한다.

3.1 제안하는 이동객체 트래킹 기법의 구조

SigMO-TRK의 전체구조는 센서노드의 배치를 위한 센서네트워크의 구성과 효율적인 이동객체의 이동정보의 집계를 위한 지역적 라우팅 계층 트리 구성으로 이루어진다. SigMO-TRK의 센서네트워크는 추적하고자 하는 전체 영역을 고정크기의 n*n 그리드 셀로 분할하여 구성한다. 그림 3은 SigMO-TRK의 센서 네트워크 구성을 나타낸다. 센서네트워크는 노드의 기능에 따라 감지노드,

싱크노드, 계층노드로 이루어진다. 감지노드는 이동객체의 이동을 감지하는 기능을 하며, 싱크노드는 감지노드로부터 감지된 이동객체의 정보가 최종적으로 집계되는 노드로써, 이동객체의 트래킹을 수행한다. 계층노드는 1개의 싱크노드를 포함하며, 감지노드로부터 감지된 이동객체의 정보를 싱크노드까지 전송하여 최종적으로 집계하는 기능을 수행한다.

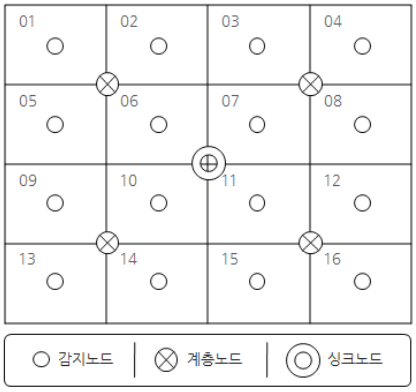


그림 3. SigMO-TRK의 센서 네트워크 구성

표 1은 SigMO-TRK의 센서네트워크 크기에 따른 필요 노드의 수를 나타낸다. SigMO-TRK는 센서네트워크 크기 n*n에서 n*n개의 감지노드를 통해 이동객체의 이동을 감지하며, 개의 계층 수 구성하여 개의 계층노드를 통해 싱크노드까지 이동정보를 전송한다. 예를 들면, 센서네트워크를 4*4 그리드 셀로 구성할 경우, 2개의 중간 계층이 생성되며 16개의 감지노드, 싱크노드를 포함한 5개의 계층노드가 필요하다. 이는 최하단의 계층인 감지노드에서 계층노드를 이용하여 2번의 통신 횟수로 싱크노드까지 이동객체의 이동정보를 전송할 수 있다.

표 1. SigMO-TRK의 센서네트워크 크기에 따른 필요 노드의 수

센서 네트워크 크기	계층수(d)	센서 노드 수	계층 노드 수
n * n	$\log_2 n$	n * n	$\sum_{i=0}^{d-1}(2^i)^2$

한편 SigMO-TRK는 감지노드에서 싱크노드로 효율적인 데이터 집계 및 전송을 위해 지역적 라우팅 계층 트리를 구성한다. 그림 4는 SigMO-TRK의 지역적 계층 트리 구성을 나타낸다. 이는 지역적으로 인접한 4개의 자식노드가 1개의 부모노드를 가지도록 구성한다. 이는, 지역을 담당하는 계층 노드를 두어 노드의 작업을 지역적으로 분산시켜 에너지의 효율성을 높인다. 특정 지역을 반복적으로 방문할 시, 공간 필터링을 통해 불필요한 메시지를 줄인다. 또한 고장 발생 시 해당지역을 제외한 지역이 문제없이 동작하도록 한다. 예를 들면, 지역적으로 인접한 01, 02, 05, 06번의 셀의 감지노드는 지역의 중간에 위치한 계층노드를 부모노드로 설정한다. 또한 4개의 인접한 계층노드는 중간에 위치한 싱크노드를 부모노드로 설정한다. 이를 통해 01, 02, 05, 06번의 셀에서 감지된 이동객체의 이동정보는 설정된 부모노드인 계층노드를 통해 싱크노드까지 전송한다. 지역적 계층 라우팅 트리의 구축 알고리즘은 그림 5와 같다. 첫째, 센서네트워크의 그리드 크기에 따라 필요한 노드 및 계층의 depth를 계산한다. 둘째, 각 depth에 속한 노드를 통해서 싱크노드 계층부터 4개의 자식노드를 할당한다. 계층의 각 노드와 할당된 자식 노드 간에 부모-자식 관계를 설정한다. 마지막으로 모든 계층의 모든 노드가 부모-자식 관계가 설정될 때까지 반복한다.

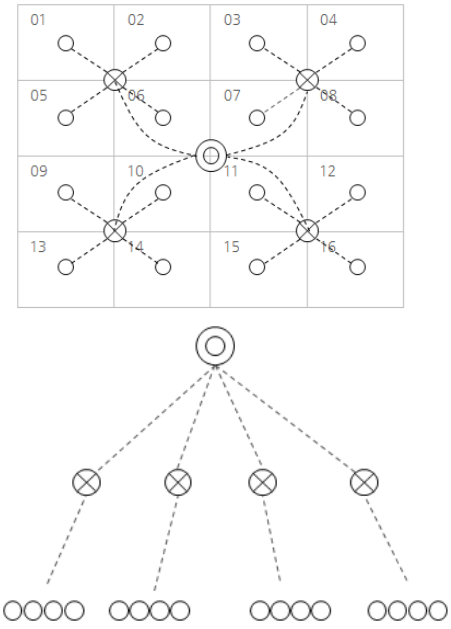


그림 4. SigMO-TRK의 지역적 계층 트리 구성

```
Algorithm Build_LocalRoutingTree(int n, int i)
Array_ChildNode[i] = ∅; // i = the maximum number of child node
01. N = Calculate_Node(n, d);
02. d = Calculate_depth(n); // log2n
03. for each depth d
04.     for each Node n ∈ Get_NodesOfDepth(d)
05.       Array_ChildNode = Allocate_ChildNode(d, 4);
06.         Set_RelationShip(n, Array_ChildNode);
End Algorithm
```

그림 5. 지역적 계층 라우팅 트리의 구축 알고리즘

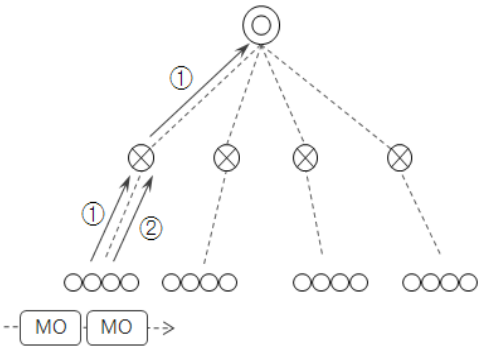
또한, 지역적 계층 라우팅 트리의 각 계층노드는 그림 6과 같은 데이터 구조를 유지한다. 첫째, MOID는 센서 네트워크 상을 움직이는 이동객체의 아이디를 나타내며, 이를 통해 다수의 이동객체를 동시에 처리하는 것이 가능하다. 둘째, signature는 하위 계층 노드로부터 집계된 이동객체의 이동정보를 나타내며, 이를 통해 노드의 재방문을 최소화하는 효율적인 트래킹 처리를 수행할 수 있다. 셋째, moved_flag는 이동객체가 현재의 계층노드를 지나갔는지 여부를 나타내며, 이를 이용한 공간 필터링 기법으로 상위 계층노드로의 반복적인 메시지를 필터링 한다. 마지막으로 aggregated_flag는 현재의 계층노드의 하위 i개의 자식노드의 signature 정보가 현재노드로 집계가 되었는지 여부를 나타내며, 하위노드의 집계가 필요할 시 이를 통해 하위노드의 signature 정보를 최종적으로 집계할 수 있다.

MOID	signature	moved_flag	aggregated_flag

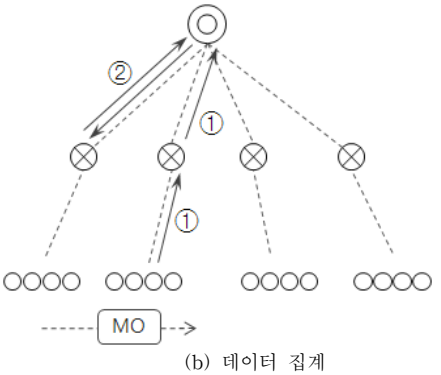
그림 6. SigMO-TRK의 계층노드의 데이터 구조

3.2 공간 필터링 기법을 이용한 데이터 집계

본 절에서는 지역적 라우팅 계층 트리를 이용한 데이터 집계에 대해 서술한다. 센서네트워크의 감지노드가 이동객체의 이동을 감지하였을 경우, 감지노드는 이동객체의 이동정보를 SigMO-TRK의 지역적 계층 라우팅 트리를 이용하여 싱크노드까지 전송한다. 그러나 이동객체가 동일한 공간을 이동할 경우, 반복적으로 싱크노드로 데이터를 전송하는 것은 에너지 낭비를 초래한다. 따라서 SigMO-TRK의 지역적 계층 라우팅 트리는 계층노드의 데이터 구조에 moved_flag 변수를 둔다. 이를 통해, 이동객체가 동일 지역을 반복적으로 이동하였을 경우, 공간 필터링을 수행하여 불필요한 메시지 전송을 차단한다. 여기서 공간 필터링이란, 해당 노드에서 상위계층 노드로 데이터 전송을 차단하는 것을 의미한다. 그림 7은 지역적 계층 라우팅 트리의 공간 필터링 기법 및 데이터 집계를 나타낸다. SigMO-TRK 의 지역적 라우팅 계층트리를 이용한 공간 필터링 기법은 다음과 같다. 첫째, 이동객체가 해당지역에 처음으로 방문할 경우, 지역적 라우팅 계층 트리를 통해서 싱크노드까지 이동정보를 전송한다. 또한, 각 계층노드는 moved_flag를 ALREADY_ MOVED로 설정하여, 이동객체가 이미 방문하였음을 알린다. 이후, 이동객체가 동일한 지역을 반복적으로 방문 시, 해당 계층 노드는 공간 필터링을 수행하며 하위 노드로부터 갱신된 이동 정보만을 현재 노드에서 집계한다. 그림 7의 (a)는 SigMO-TRK의 지역적 라우팅 계층트리의 공간필터링 기법을 나타낸다. ①은 이동객체가 해당지역을 처음으로 방문할 경우의 데이터 전송을 나타내며, ②는 이동객체가 동일한 지역을 반복적으로 방문 시의 공간 필터링을 나타낸다. 그림 7의 (b)는 SigMO-TRK의 지역적 라우팅 계층트리의 데이터 집계를 나타낸다. ①은 이동객체가 다른 지역으로 이동하였을 경우의 데이터 전송을 나타내며, ②는 싱크노드가 객체의 이동이 발생한 첫번째 지역의 데이터를 요청하고 최종적으로 데이터를 집계하는 것을 나타낸다. 따라서 SigMO-TRK는 공간 필터링 및 데이터 집계를 통해 싱크노드에서 최신의 이동객체의 궤적 정보를 유지할 수 있다. 이를 통해 이동객체 트래킹을 싱크노드에서 즉시 처리할 수 있다.



(a) 공간 필터링 기법



(b) 데이터 집계

그림 7. 지역적 계층 라우팅 트리의 공간 필터링 기법 및 데이터 집계

3.3 시그니처를 이용한 이동객체 트래킹

SigMO-TRK의 이동객체 트래킹은 이동객체의 궤적 시그니처 구성과 이를 통한 이동객체 트래킹으로 이루어진다. SigMO-TRK의 감지노드는 이동객체의 이동정보를 시그니처로 구성하며, 이동정보인 시그니처는 지역적 라우팅 계층트리를 통해 싱크노드로 집계되어 최종적으로 이동객체의 궤적 시그니처를 구성한다. 이 후 집계된 이동객체의 궤적 시그니처를 통해 이동객체 트래킹을 수행한다. 그림 8은 이동객체의 궤적 시그니처의 구성을 나타낸다. 시그니처는 SigMO-TRK의 센서네트워크의 고정 크기의 그리드 셀 개수(n) 만큼의 비트 열 크기를 가진다. 각 비트 열은 각 셀 내의 이동객체의 이동 여부를 표시하며 셀 내 이동시 1로 세팅(setting)한다. 이는, 이동

객체의 궤적의 길이가 짧은 경우에 이동한 노드의 수보다 많은 그리드 셀 개수(n)만큼의 비트 열 크기를 가지만 반면에, 이동객체의 길이가 매우 길어져도 동일한 비트 수로 처리할 수 있음을 나타낸다.

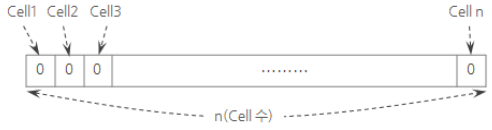
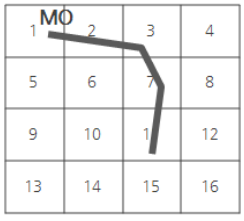
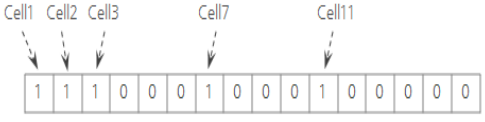


그림 8. 이동객체의 궤적 시그니처 구성

그림 9는 이동객체의 궤적 시그니처 구성의 예를 나타낸다. 그림 9의 (a)에서 이동객체 MO는 센서네트워크에서 1, 2, 3, 7, 11번 셀을 이동한다. 각 셀의 감지노드는 이동객체의 이동을 감지하였을 경우, 해당되는 비트열을 1로 세팅하여 구성하며, 각 계층노드는 감지노드로부터 시그니처를 집계하여 싱크노드로 이동객체의 최종 궤적 시그니처를 전송한다. 이를 통해 구성된 최종 이동객체의 궤적 시그니처는 그림 9의 (b)와 같다.



(a) 이동객체의 센서네트워크 내에 이동



(b) 이동객체의 궤적 시그니처

그림 9. 이동객체의 궤적 시그니처 구성의 예

SigMO-TRK에서 이동객체 트래킹은 이동객체의 궤적 시그니처 수집 및 시그니처를 통한 이동객체 트래킹으로 구성된다. 이동객체의 궤적 시그니처 수집은 감지노드와 계층노드를 통해 싱크노드까지 이동객체의 궤적 시그니처를 수집한다. 먼저, 이동객체 궤적 시그니처 수집 알고리즘은 그림 10과 같다. 이동객체 궤적 시그니처 수집은 이동객체가 방문한 노드가 처음 방문한 노드인 경우와 이동객체가 방문한 노드가 이전에 이미 방문했던 노드인 경우로 이루어진다. 먼저, 이동객체가 처음 방문한 노드인 경우는 첫째, 방문한 노드의 셀 아이디를 통해서 시그니처를 생성한다. 둘째, 생성한 시그니처 정보를 센서노드의 Signature 변수에 대입한다. 셋째, 센서노드의 moved_flag 변수를 ALREADY_MOVED로 설정하여 이미 방문했음을 표시한다. 마지막으로 상위 센서노드로 시그니처 정보와 함께 데이터를 전송한다. 데이터를 전송받은 상위노드는 MO_Signature_Aggregation 함수를 재수행하여 데이터를 집계한다. 한편, 이동객체가 방문한 노드가 이전에 이미 방문했던 노드인 경우는 다음과 같이 수행된다. 첫째, 방문한 센서노드의 시그니처 정보인 Signature에 하위노드로부터 받은 시그니처와 OR연산을 통해 시그니처를 집계한다. 둘째, 방문 센서노드의 하위노드의 시그니처 수집 여부 플래그인 aggregated_flag를 확인하여 수집되지 않은 하위노드로 Get_Signature 메시지를 전송하여 하위노드로부터 시그니처를 요청한다. 넷째, 하위노드로부터 수신한 시그니처 정보를 현재의 시그니처와 OR연산을 통해 시그니처를 집계한다. 이러한 방법을 통해 싱크노드까지 이동객체의 궤적 시그니처가 수집된다. 마지막으로, 이동객체 트래킹은 감지노드의 재방문 없이 이동객체의 궤적 시그니처를 통해 싱크노드에서 직접 처리한다.

```
Algorithm MO_Signature_Aggregation(visited_Cell, signature)
// visited_Cell : 이동객체가 방문한 셀의 노드, signature : 하위노드로부터 수신한 signature
01. if(visited_Cell.moved_flag == NOT_MOVED) {
02. MO_Signature = Make_Signature(visited_Node.id);
03.   visited_Cell.Signature = MO_Signature;
04.   visited_Cell.moved_flag = NOT_MOVED;
05.   Send_Data(Upper_Node, MO_Signature);
06. }
07. else if (visited_Cell.moved_flag ==
            ALREADY_MOVED) { // pruned
08.    visited_Cell.Signature |= MO_Signature;
09.    for(i = 0; i < count_bottomNode; i++) {
10.     if(aggregated_flag[i] == 0){
11.       Send_Request(Bottom_Node, Get_Signature);
12.       bottomNode_Signature = Get_Response();
13.       visited_Cell.Signature |=
                bottomNode_Signature;}
14.    } // for
15. } //else if
End Algorithm
```

그림 10. 이동객체 궤적 시그니처 수집 알고리즘

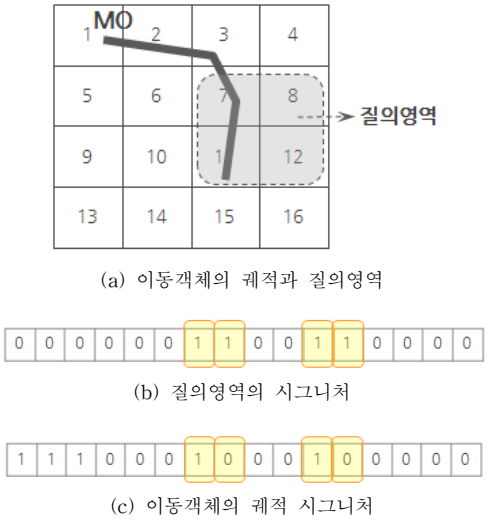
싱크노드에 수집된 이동객체의 트래킹 처리 알고리즘은 그림 11과 같다. 첫째, 이동객체가 마지막으로 지나간 이동객체의 시그니처는 집계가 되지 않았기 때문에 싱크노드의 completed_flag 정보를 확인하여 해당 노드로 이동객체의 시그니처의 최종 집계를 요청한다. 둘째, 유저로부터 트래킹 하고자 하는 질의영역의 셀 아이디 리스트를 받는다. 셋째, 싱크노드가 유지하고 있는 이동객체의 궤적 시그니처를 얻는다. 넷째, 트래킹 하고자 하는

질의 영역의 셀 아이디로 부터 질의 시그니처를 생성한다. 마지막으로, 싱크노드가 유지하고 있는 이동객체의 궤적 시그니처와 질의 시그니처와의 비교를 통해 이동객체가 지나간 셀을 찾은 후 이동객체 트래킹 알고리즘을 종료한다.

```
Algorithm MO_Tracking(qcell_IdList) //qcell_IdList : 질의 셀 아이디 리스트
01.qcell_SigList = ∅
02.resultSet = ∅
03. last_Aggregation();
04.MO_Signature = getSignature_MO( );
05.while(qcell_IdList != NULL) {
06.   temp_Sig = make_signature(qcell_IdList[i].id);
07.   qcell_SigList.Add(temp_Sig);
08.   cnt++;
09.}
10.for(i=0; i<cnt; i++) {
11.   if(MO_Signature & qcell_SigList[i] ==  qcell_SigList[i])
12.      ResultSet.Add(qcell_IdList[i].id);
13.}
End Algorithm
```

그림 11. 이동객체의 트래킹 처리 알고리즘

그림 12는 이동객체 트래킹 처리의 예를 나타낸다. 그림 12의 (a)는 이동객체의 궤적과 질의 영역을 나타낸다. 이동객체의 궤적은 1, 2, 3, 7, 11번 셀을 이동하였으며, 질의영역은 7, 8, 11, 12번 셀로 설정하였다. 즉, 이동객체가 질의영역인 7, 8, 11, 12번 셀 영역을 지나갔는지를 트래킹 한다. 그림 12의 (b)는 질의영역을 통해 생성한 질의 시그니처를 나타내며, 그림 12의 (c)는 이동객체의 궤적 시그니처를 나타낸다. 싱크노드는 질의 시그니처의 1로 세팅된 비트열과 이동객체의 궤적 시그니처의 동일 비트 열과 비트 AND 연산을 통해 이동객체 트래킹을 수행한다. 그 결과 이동객체의 궤적은 질의영역중 7, 11번 셀을 지나갔다는 것을 알 수 있다.



(a) 이동객체의 궤적과 질의영역

(b) 질의영역의 시그니처

(c) 이동객체의 궤적 시그니처

그림 12. 이동객체 트래킹 처리의 예

4. 시그니처를 이용한 이동객체 유사궤적 질의처리

센서 네트워크에서 유사궤적의 질의처리는 이동객체들의 궤적을 수집하여 현재 및 미래를 위한 지식을 도출할 수 있는 응용에 쓰일 수 있다. 예를 들면, 정체가 심한 지역에서 자동차들의 궤적을 수집한 후, 유사궤적 질의처리를 통해 정체 해소를 위한 신규 도로 개설 등에 사용될 수 있다. 본 절에서는 SigMO-TRK를 확장하여 다수의 이동객체의 궤적 시그니처를 수집한 후, 질의궤적과 유사한 궤적을 찾는 유사궤적 질의처리 기법을 기술한다.

4.1 SigMO-TRK의 확장

이동객체 트래킹을 위한 SigMO-TRK는 계층 노드에서 시그니처가 수집되지 않은 노드에게 시그니처 집계를 요청하는 방식을 통해 최종 궤적 시그니처를 구성한다. 이러한 구조는 다수의 이동객체의 궤적을 동시에 처리하는데 있어, 센서노드의 저장 공간의 한계로 인해 비효율적이다. 따라서 다수의 이동객체의 궤적을 통한 유사궤적 질의처리를 위해서 SigMO-TRK 구조의 확장이 필요하다. 따라서 본 논문에서는 유사궤적 질의처리를 위한 확장된 SigMO-TRK의 구조를 제시하며, 그 특징은 다음과 같다. 첫째, SigMO-TRK의 각 계층노드의 기존 데이터 구조에 이동객체의 이동 순서를 저장하기 위한 step 변수를 추가한다. 이를 통해, 싱크노드에서 유사궤적질의처리 시 노드의 재방문 없이 정확한 궤적 비교를 수행할 수 있다. 둘째, 이동객체 궤적 시그니처 및 이동 순서 정보를 싱크노드로 수집할 때, 수집되지 않은 하위 노드로부터의 요청 뿐만 아니라 센서노드에 유지하고 있는 데이터양이 일정 Threshold 이상일 경우 자동적으로 상위 부모노드로 데이터를 전송하도록 확장한다. 이를 통해 센서노드의 저장 공간을 효율적으로 이용할 수 있다.

4.2 유사궤적 질의처리 알고리즘

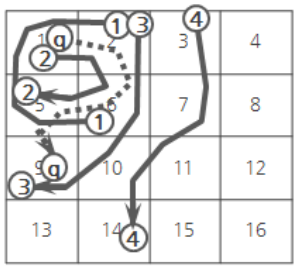
이동객체의 트래킹 기법에서의 유사궤적 질의처리 방법은 싱크노드에서 저장하고 있는 이동객체의 궤적 시그니처를 통해 처리한다. 시그니처는 일정크기의 비트열로 구성되어 있기 때문에, SigMO-TRK는 시그니처를 이용하여 유사정도를 측정하기 위해 유사 허용 비트수(Threshold)를 설정한다. 따라서 질의 시그니처와 이동객체의 궤적 비교 결과 유사하지 않는 비트의 수가 유사 허용 비트수보다 작다면 유사한 궤적의 후보 셋으로 선

정한다. 그림 13은 SigMO-TRK의 유사궤적 질의처리 알고리즘을 나타낸다. 첫째, 유저에게 질의 궤적 셀 아이디 리스트와 유사 허용 비트수를 입력받는다. 둘째, 입력받은 질의 궤적 셀 아이디 리스트를 통해 질의 궤적의 시그니처를 생성한다. 셋째, 이동객체의 궤적 시그니처가 집계되지 않은 노드에 대해서 싱크노드로 이동객체의 궤적의 집계를 수행한다. 넷째, 질의 이동객체 시그니처와 질의 시그니처와의 시그니처 비교를 수행한다. 다섯째, 입력받은 유사 허용 비트수를 넘지 않으면 이동객체의 궤적에 대해서 유사궤적의 후보 셋으로 선정한다. 마지막으로, 선정된 후보 셋의 이동객체의 궤적과 질의 궤적의 이동객체의 이동순서 비교를 통해 최종 유사한 궤적을 선정한다.

```
Algorithm MO_SimilarTrajectory(qcell_IdList, threshold)
//threshold : 유사 허용 비트수
01. CandidateSet = ∅
02. ResultSet = ∅
03.while(qcell_IdList != NULL){
04.   temp_Sig = make_signature(qcell_IdList[i].id);
05.   qcell_SigList.Add(temp_Sig);
06.   cnt++;
07.}
08.MOs_Signature = getSignature_MOs( );
09.while(MOs_Signature != NULL) {
10.   count = CompareSignature(qSignature);
11.   if(count <= threshold)
12.          CandidateSet.Add(MOs_Signature);
13.   MOs_Signature = MOs_Signature->next;
14.} // while
15.for each MO in CandidateSet {
16.     count = CompareStep(MO.step);
17.     if(count <= threshold)
18.         ResultSet.Add(MO);
19.}
End Algorithm
```

그림 13. SigMO-TRK의 유사궤적 질의처리 알고리즘

그림 14는 유사궤적 질의처리의 예를 나타낸다. 그림 14의 (a)는 각 이동객체 및 질의 궤적을 나타낸다. ①번 이동객체는 2, 1, 5, 6번 셀을 순서대로 이동하였으며, ②번 이동객체는 1, 2, 6, 5번 셀을 순서대로 이동하였다. ③번 이동객체는 2, 6, 10, 9번 셀을 순서대로 이동하였고, ④번 이동객체는 3, 7, 11, 10, 14번 셀을 순서대로 이동하였다. 마지막으로 질의궤적은 1, 2, 6, 5, 9번 셀을 순서대로 지나간 궤적으로 설정하였다. 즉 질의궤적인 1, 2, 6, 5, 9번의 궤적과 유사한 이동객체의 궤적을 검색한다. 그림 14의 (b)는 이동객체 및 질의 궤적의 시그니처를 나타내며, 그림 14의 (c)는 유사비트 허용수인 Threshold가 2인 경우의 (b)의 이동객체 궤적과 질의 궤적 간의 시그니처 비교를 통해 얻어진 유사궤적 후보 집합이다. 즉 ④번 이동객체는 유사비트 허용수인 2를 초과하였기 때문에 후보 집합에서 제외된다. 즉, 시그니처 비교를 통한 필터링으로 유사궤적 비교를 위한 궤적의 수를 줄인다. 그림 14의 (d)는 질의 궤적의 이동 순서와 후보 집합의 이동객체의 이동순서를 비교하여, threshold를 초과하지 않는 최종 유사궤적 결과 집합을 나타낸다. 즉, 이동객체 ②, ③번 이동객체가 질의궤적과 유사한 궤적임을 알 수 있다.



(a) 이동객체 및 질의 궤적

q	1	1	1	0	0	1	0	0	1	0	0	0	0	0	0	0
①	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0
②	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0
③	0	1	0	0	0	1	0	0	1	1	0	0	0	0	0	0
④	0	0	1	0	0	0	1	0	0	1	1	0	0	1	0	0

(b) 이동객체 및 질의 궤적 시그니처

①	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0
②	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0
③	0	1	0	0	0	1	0	0	1	1	0	0	0	0	0	0

(c) 시그니처 비교를 통한 후보 유사궤적(Threshold : 2)

②	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0
③	0	1	0	0	0	1	0	0	1	1	0	0	0	0	0	0

(d) 방문순서 비교를 통한 최종 유사궤적(Threshold : 2)

그림 14. 이동객체 트래킹 처리의 예

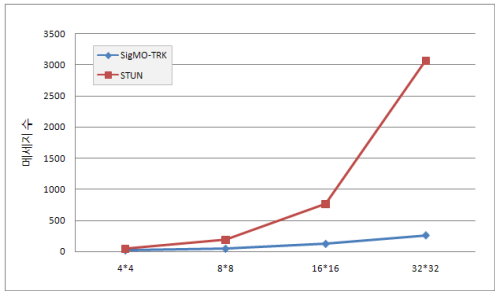
5. 성능평가

본 절에서는 제안하는 시그니처를 이용한 이동객체 트래킹 기법인 SigMO-TRK의 성능 평가를 수행한다. 성능 비교 대상은 기존의 대표적인 이동객체 트래킹 기법인 STUN 이다. DIM은 트래킹 계층 트리의 구조 변경이 매우 빈번하게 필요하며, STUN에 비해 성능이 좋지 않기 때문에 성능평가 대상에서 제외하였다[8,10]. 성능평

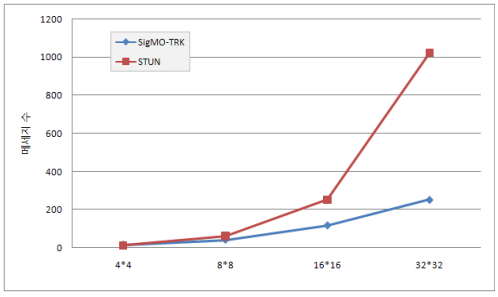
가 항목은 첫째, 센서네트워크 크기에 따른 트래킹 처리 시간, 둘째, 질의영역 크기에 따른 트래킹 처리 시간, 마지막으로 센서네트워크 크기에 따른 유사궤적 질의처리 성능 이다. 성능 평가 환경은 인텔 Core2 Quad CPU, 2G 메인 메모리에서 TinyOS 1.x[14]의 TOSSIM 시뮬레이터[15]를 이용하여 구현 및 성능평가를 수행하였다. 아울러, 성능 평가를 위해 각 센서네트워크 그리드 셀 크기는 4*4, 8*8, 16*16, 32*32로 설정하여 감지노드를 배치하였고, 궤적의 구성하는 노드의 수는 10, 15, 20, 30으로 설정하였다. 센서네트워크 크기에 따른 SigMO-TRK와 STUN의 구성을 위해 필요한 노드의 개수는 <표 2>와 같다.

5.1 센서네트워크 크기에 따른 트래킹 처리

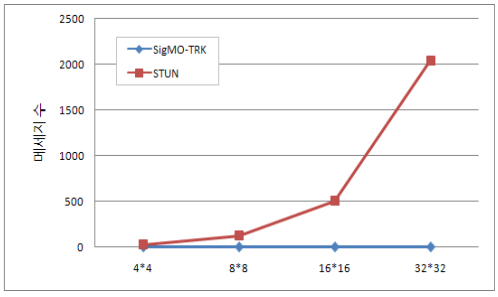
성능 평가 항목은 이동객체의 궤적의 길이를 10으로 고정하고, 센서 네트워크 그리드 셀 크기를 각각 4*4, 8*8, 16*16, 32*32로 증가하면서 각각의 메시지수를 측정한다. 그림 15의 (a)는 센서네트워크 크기 변화에 따른 트래킹 처리를 위한 전체 메시지 수를 나타낸다. 트래킹 처리를 위한 전체 메시지 수는 그림 15의 (b)의 데이터 집계를 위한 메시지 수와 그림 15의 (c)의 트래킹 처리를 위한 메시지 수의 합이다. 그림 15의 (a)에서처럼 센서네트워크 크기가 증가할수록 두 방법 모두 전체 메시지 수가 증가함을 알 수 있다. 이는 센서네트워크 크기가 증가할수록 싱크노드까지 데이터를 전송하기 위한 중간 계층 노드의 수가 증가하기 때문이다. 또한 센서 네트워크 크기가 8*8 이상일 경우, STUN의 메시지 수가 SigMO-TRK보다 더욱 증가함을 알 수 있다. 이는 그림 15의 (b)와 같이 중간 계층 노드의 증가로 인해 데이터 집계를 위한 메시지 수가 증가하며, 그림 15의 (c)와 같이 트래킹 처리를 위한 노드 재방문을 위해 많은 메시지 수가 발생된다. 반면에 SigMO-TRK는 STUN보다 지역적 라우팅 계층 트리를 통해 적은 중간 계층노드의 개수 및 트래킹 처리를 위해 이동객체가 마지막으로 이동한 노드만의 재방문(2회의 메시지 수)으로 트래킹을 처리할 수 있기 때문이다. 이를 통해, STUN보다 적은 메시지 수로 우수한 성능을 나타낸다.



(a) 전체 메시지 수



(b) 데이터 집계를 위한 메시지 수



(c) 트래킹 처리를 위한 메시지 수

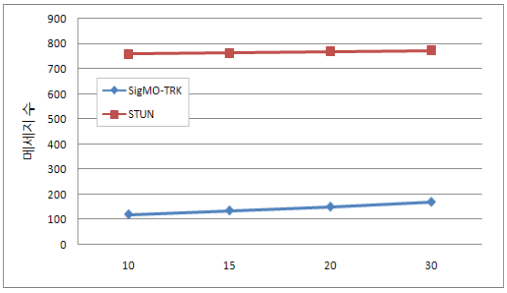
그림 15. 센서네트워크 크기 변화에 따른 트래킹 처리 성능

표 2. 센서네트워크 크기에 따른 SigMO-TRK 및 STUN 구성을 위한 노드 개수

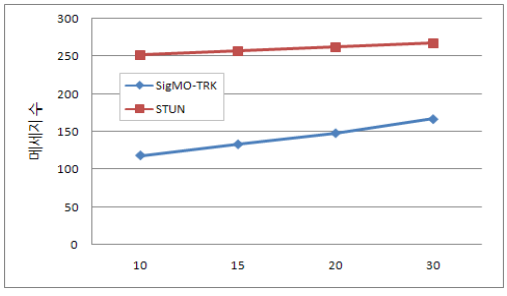
	4*4			8*8		
	감지노드수	계층노드수	전체노드수	감지노드수	계층노드수	전체노드수
SigMO-TRK	16	5	21	64	21	85
STUN	16	15	31	64	63	127
	16*16			32*32		
	감지노드수	계층노드수	전체노드수	감지노드수	계층노드수	전체노드수
SigMO-TRK	256	85	341	1024	341	1365
STUN	256	255	511	1024	1023	2047

5.2 이동객체 궤적의 길이에 따른 트래킹 처리

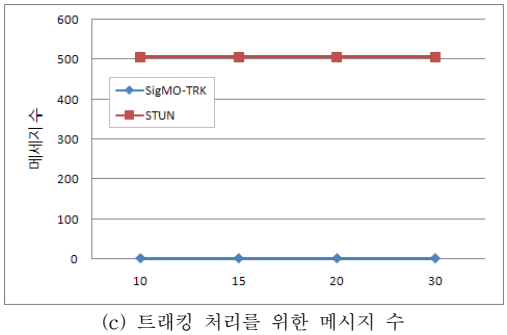
성능 평가 항목은 고정된 16*16 크기의 센서네트워크에서 이동객체의 궤적의 길이 즉, 방문 노드의 수를 각각 10, 15, 20, 30으로 증가하면서 각각의 메시지수를 측정하였다. 그림 16 (a)는 이동객체의 궤적의 길이 변화에 따른 트래킹 처리를 위한 전체 메시지 수를 나타낸다. 트래킹 처리를 위한 전체 메시지 수는 그림 16 (b)의 데이터 집계를 위한 메시지 수와 그림 16의 (c)의 트래킹 처리를 위한 메시지 수의 합이다. 그림 16의 (a)에서처럼 질의영역 크기가 증가할수록 두 방법 모두 메시지 수가 증가하는 것을 알 수 있다. 이는 이동객체의 궤적의 길이가 변화하면서 보다 많은 감지노드를 방문을 통해 싱크노드까지 전송을 위한 더 많은 메시지가 발생하기 때문이다. 그러나 SigMO-TRK는 공간 필터링 기법을 통해



(a) 전체 메시지 수



(b) 데이터 집계를 위한 메시지 수



(c) 트래킹 처리를 위한 메시지 수

그림 16. 이동객체의 궤적의 길이에 따른 트래킹 처리 성능

이동객체의 궤적의 길이가 길어져도 중간 계층노드에서 데이터를 집계한 후 상위노드로의 전송하기 때문에 선형적으로 메시지의 수가 증가한다. 그림 16의 (c)에서는 이동객체의 궤적의 길이가 증가하여도, 두 방법의 트래킹 처리를 위한 메시지 수는 일정함을 나타낸다. 이는 STUN의 경우는 트래킹 처리를 위해서 이동객체 궤적의 길이에 관계없이 16*16 크기의 센서 네트워크에서 동일한 노드의 재방문(506회의 메시지 수)이 필요하기 때문이며, SigMO-TRK는 이동객체 트래킹을 위해서 이동객체가 마지막으로 이동한 노드만의 재방문(2회의 메시지 수)으로 이동객체 전체 궤적의 시그니처를 집계할 수 있기 때문이다. 결과적으로, 데이터 집계시 SigMO-TRK는 지역적 라우팅 계층 트리에 의한 공간필터링 및 트래킹 처리를 위한 최소한의 노드만을 재방문하기 때문에, STUN보다 우수한 성능을 나타낸다.

5.3 센서네트워크 크기에 따른 유사궤적 질의처리

본 절에서는 다수의 이동객체의 궤적을 통해 질의궤적과 유사한 궤적을 검색하는 유사궤적 질의처리 성능평가를 수행한다. STUN은 다수의 이동객체를 동시에 처리할 수 없기 때문에 유사궤적 질의를 지원하지 못한다. 따라서 개별적으로 SigMO-TRK의 유사궤적 질의 처리의 성능을 평가한다. 먼저, 이동객체의 수를 10으로 고정시킨 다음, 센서 네트워크 그리드 셀 크기를 각각 4*4, 8*8, 16*16, 32*32로 증가하면서 메시지수를 측정하였다. 그림 17은 유사궤적 질의 처리 시의 메시지 개수를 나타내며, 센서네트워크의 크기에 관계없이 일정한 메시지 수(20회의 메시지 수)가 요구됨을 알 수 있다. 이는 유사궤적 질의처리도 이동객체 트래킹과 동일하게 싱크노드에 집계된 이동객체의 시그니처를 통해 수행하기 때문이다. 즉, 이동객체가 마지막으로 이동한 노드만의 재방문을 위해 2회의 메시지 수가 필요하며, 이동객체별로 동일하게 수행하므로, 이동객체의 수가 10개의 경우 총 20번의 메시지가 소요된다. 따라서 SigMO-TRK는 유사궤적 질의를 센서네트워크의 크기에 관계없이 수행하며, 이동객체의 수의 2배의 메시지 횟수로 효율적으로 처리할 수 있음을 나타낸다.

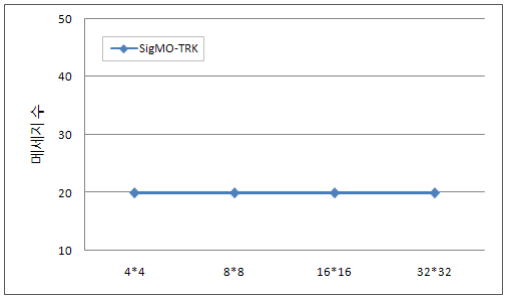


그림 17. 센서네트워크 크기에 따른 유사궤적 질의처리

6. 결론 및 향후연구

현재 유비쿼터스 센서 네트워크 기술에 대한 관심이 크게 고조되고 있으며 또한, 센서노드의 하드웨어 기술 발전으로 인해 확장된 저장 공간에서 복잡한 연산을 처리하는 것이 가능하게 되었다. 이를 통해 다양한 분야에 센서네트워크 기술을 적용한 많은 응용서비스가 개발되고 있으며, 센서 네트워크의 응용분야 중에서 이동객체의 이동경로를 트래킹(tracking) 하는 이동객체 트래킹 기법이 중요한 이슈 중 하나이다.

기존의 이동객체 트래킹 기법은 센서노드의 재방문을 통해 이동여부를 확인하기 때문에, 이로 인한 많은 메시지 수로 인해 에너지 비효율성을 초래한다. 따라서 본 논문에서는 시그니처를 이용한 효율적인 이동객체 트래킹 기법(SigMO-TRK)을 제안하였다. 제안하는 기법은 공간 필터링을 적용한 지역적 라우팅 계층트리를 통해 효율적으로 이동객체의 이동 정보를 집계하며, 센서노드에서 저장하고 있는 이동객체의 이동정보를 시그니처로 구성하여, 효율적으로 이동객체의 트래킹을 처리하였다. 또한, SigMO-TRK의 구조를 확장하여 다수의 이동객체들의 궤적 시그니처를 통해 질의 궤적과 유사한 이동객체의 궤적을 검색하는 유사궤적질의를 수행하였다. 또한, 센서네트워크에서 이동객체 트래킹을 위한 대표적인 연구인 STUN과의 성능 비교를 통하여 이동객체의 데이터 집계 및 트래킹 처리 메시지의 수가 현저히 감소됨을 알 수 있었다. 또한 다수의 이동객체의 지원을 통하여 이동객체 궤적들의 유사궤적을 적은 수의 메시지를 통해 효율적으로 처리할 수 있음을 제시하였다.

향후 연구로는 본 논문에서 제안한 시그니처를 이용한 이동객체 트래킹 기법인 SigMO-TRK를 실제 센서를 이용한 차량 위치 추적 시스템과 같은 이동객체 트래킹 분야의 응용에서 그 효율성을 입증하는 것이다.

참 고 문 헌

[1] W. Zhang and G. Cao, “DCTC: Dynamic Convoy Tree-Based Collaboration for Target Tracking in Sensor Networks,” IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, Vol. 3, No. 5, 2004, pp. 1689-1701.

[2] X. Ji, H. Zha, J. J. Metzner, and G. Kesidis, “Dynamic cluster structure for object detection and tracking in wireless ad-hoc sensor networks,” IEEE Int'l. Conf. on Communications, Vol. 7, 2004, pp. 3807-3811.

[3] Y. Xu, J. Winter, W. C. Lee “Prediction-based strategies for energy saving in object tracking sensor networks,” IEEE Int'l. Conf. on Mobile Data Management, 2004, pp. 346-357.

[4] A. Meka, A. K. Singh, “DIST : A Distributed Spatio-temporal Index Structure for Sensor Networks,” Proc. of the 14th ACM Int'l. Conf. on Information and Knowledge Management. 2005, pp. 139-146.

[5] 강홍구, 박치민, 홍동숙, 한기준, “공간 센서 데이타의 효율적인 실시간 처리를 위한 공간 DSMS의 개발,” 한국공간정보시스템학회 논문지, Vol.9, No.1, 2007, pp.45-57.

[6] J. Aslam, Z. Butler, F. Constantin, V. Crespi, G. Cybenko, and D. Rus, “Tracking a Moving Object with Binary Sensors,” Proc. of ACM SenSys, 2003, pp. 150-161.

[7] K. Mechitov, S. Sundresh, Y. Kwon, and G. Agha, “Cooperative Tracking with Binary-Detection Sensor Networks,” Tech. Rep. UIUCDCS-R-2003-2379, University of Illinois at Urbana-Champaign, 2003.

[8] X. Li, Y. Kim, R. Govindan, and W. Hong. "Multi-dimensional range queries in sensor networks," Proc. of ACM SenSys, 2003, pp. 63-75.

[9] B. Greenstein, D. Estrin, R. Govindan, S. Ratnasamy, and S. Shenker, "DIFS:A distributed index for features in sensor network," Proc. of IEEE International Workshop on Sensor Network Protocols and Applications, 2003. pp. 163-173.

[10] H. T. Kung and D. Vlah, “Efficient Location Tracking Using Sensor Networks,” Proc. of IEEE Wireless Communications and Networking Conference, Vol. 3, 2003, pp. 1954-1961.

[11] 이연, 어상훈, 조숙경, 이순조, 배해영, “USN환경에서 효율적인 공간영역질의를 위한적응형 영역 집계 인덱스 기법,” 한국공간정보시스템학회 논문지, Vol. 9, No. 2, 2007, pp.93-107.

[12] B. Karp and H. T. Kung. “GPSR: Greedy Perimeter Stateless Routing for Wireless Networks,” Proc. of the Sixth Annual Int'l. Conf. on Mobile Computing and Networking, 2000, pp. 243-254.

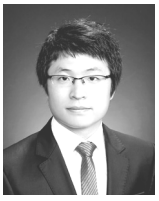
[13] J. Zobel, A. Moffat, and K. Ramamohanarao, “Inverted Files versus Signature Files for Text Indexing,” ACM Trans. on Database Systems, Vol. 23, No.4, 1998, pp. 453-490.

[14] http://www.tinyos.net/tinyos-1.x/doc/tutorial/lesson5.html

[15] P. Levis, N. Lee, M. Welsh, and D. Culler, “TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications”, Proc. of the 1st Conf. on Embedded Networkd Sensor Systems, 2003, pp. 126-137.

김 용 기

2002년 전북대학교 컴퓨터공학과 (학사)
2005년 전북대학교 대학원 컴퓨터공학과 (석사)
2006년~현재 전북대학교 대학원 컴퓨터공학과 박사과정
관심분야 : 공간 네트워크 데이터베이스, 질의처리 알고리즘, 센서 네트워크



김 영 진

2007년 전북대학교 컴퓨터 공학과 (학사)
2009년 전북대학교 대학원 컴퓨터공학과 (석사)
2009년~현재 : 삼성전자 DMC부문 DMC 연구소 SW솔루션팀 근무
관심분야 : LBS, UX 서비스, 센서 네트워크

윤 민

2008년 전북대학교 컴퓨터공학과 (학사)
2008년~현재 전북대학교 컴퓨터공학과 석사과정
관심분야 : 센서 네트워크, 공간 데이터베이스, 고차원 데이터 처리 기술

장 재 우

1984년 서울대학교 전자계산기공학과 (학사)
1986년 한국과학기술원 전산학과(석사)
1991년 한국과학기술원 전산학과(박사)
1996년~1997년 Univ. of Minnesota, Visiting Scholar
2003년~2004년 Penn State Univ., Visiting Scholar
1991년~현재 전북대학교 전기전자컴퓨터공학부 교수
관심분야 : 공간 네트워크 데이터베이스, 정보보호, 하부저장구조, 센서 네트워크