제22회 한국정보처리학회 추계학술발표대회 논문집 제11권 제2호 (2004. 11)

P2P를 이용한 이동 객체 궤적 분산 색인 방법

박경민, 강혜영 이기준
부산대학교 컴퓨터 공학과
e-mail:{kmbach, hykang, lik}@isel.cs.pusan.ac.kr

Distributed Indices of Trajectory of Moving Objects

Kyoung-Min Park, Hye-Young Kang, Ki-Joune Li
Dept. of Computer Science, Pusan National University

요 약

수십, 수백만의 이동 객체가 존재하는 환경에서 전체 이동 객체의 궤적을 중앙 서버가 모두 관리하는 것은 적절한 접근 방법이 아니다. 통신 메시지들이 서버에 집중되기 때문에, 높은 네트워크 대역폭, 처리 능력, 그리고 방대한 저장 공간을 보유한 서버를 필요로 하기 때문이다. 이에 본 논문에서는 중앙 서버없이 각각의 이동 객체들이 자기 자신의 궤적을 관리하는 방식을 통해 앞서의 문제를 해결하려 한다. 중앙 서버없이 데이터가 네트워크에 분산되어 있는 경우, 특정 데이터에 효율적으로 접근하기 위해서는 색인이 필요하게 되는데, 본 논문에서는 헤더 객체와 헤더 검색 트리라는 것을 정의하여 IPv6의 모바일 IP를 이용한 P2P방법으로 영역 질의, 최근접 질의, 궤적 질의 처리를 할 수 있는 모델을 제시한다.

1. 서론

이동 객체의 과거 궤적 데이터를 저장, 검색, 관리 하는 연구는 물류, 교통, 날씨, 군사 등 사회 전반에서 다양하게 이용되고 있다. 이러한 연구는 모두 중앙 집중화된 서버에서 이동 객체의 궤적 데이터를 관리하고 질의를 처리한다.

그러나 수십, 수백만의 이동 객체를 관리해야 하는 유비쿼터스 환경에서는 중앙의 서버가 전체 객체의 궤적 데이터를 관리하는 것이 비효율적이다. 서버가 감당하기 힘들 정도로 집중되는 질의 처리와 데이터 갱신, 그리고 통신 트래픽 등의 문제는 새로운 접근 방식을 요구한다.

그래서 본 논문에서는 각각의 이동 객체들이 자신의 궤적을 관리하는 방식을 통해 궤적 데이터를 완전 분산하고, 공간을 그리드화한 뒤 동적으로 생성되는 헤더 검색 트리와 헤더 객체라는 것을 두어 IPv6의 모바일 IP[4]로 각각의 이동 객체들이 p2p로 통신하여 질의를 처리 할 수 있는 모델을 제시한다.

본문의 내용은 다음과 같이 구성된다. 2 장에서는 분산 색인에 관련된 연구에 대해 살펴볼 것이다. 3 장에서는 본 논문이 제안하는 P2P를 이용한 이동 객체 궤적 분산 색인 모델의 구조와 원리, 질의 처리 등에 대해 알아본다. 마지막으로 4 장에서는 결론 및 향후 연구에 대해 서술한다.

2. 관련연구 및 배경

본장에서는 P2P 시스템 중 노드간의 연결이 체계적으로 형성된 구조적인 P2P 네트워크에 대한 기존 연구들에 대해 살펴본다.

분산 해쉬 테이블(DHT[1])는 해쉬 테이블을 네트워크 환경에 위치한 노드들에 분산, 저장하는 것이다. DHT를 사용하는 가장 큰 목적은 네트워크의 구조화를 촉진하여 신속하고 체계적인 검색 및 라우팅을 수행하는 것이다. 파일 공유의 경우에는 공유 네트워크를 구성하는 노드들에 DHT 버킷을 분산시킨 다음, DHT 버킷을 찾기 위해 버킷을 가진 노드를 쉽게 찾도록 한다. DHT 버킷과 노드를 연결시키기 위해 파일 이름을 해쉬한 식별자를 사용한다.

● **Chord[2]**

노드와 키의 해쉬 값을 위해 원형의 m-bit 식별자 공간을 사용한다. 키를 해쉬 값을 식별자 공간으로 대응시킬 때는 consistent hashing과 같이 원형 식별자 공간에서 해쉬된 키 값이 같거나 더 큰 노드 식별자를 같는 첫 번째 노드에 저장한다. 이 노드를 접근자(successor) 노드라 한다. 또한 분산 환경에 적용시키기 위해 finger table을 각 노드가 유지한다. 이것은 consistent hashing에서 요구되는 전체 네트워크 상의 노드에 대한 정보를 분산된 동적인 환경에서 효과적으로 만족시키기 위해 도입되었다. Finger table은 키를 삽입하거나 키를 가진 노드를 찾기 위한 lookup 등의 메시지를 해당 노드로 전달하는데 이용된다.

● **CAN[3]**

CAN에서 데이터는 유일 키를 해쉬 함수를 통해 d차원에서 한 점에 대응되는 벡터 p(p=hash(key))를 얻는다. CAN은 d차원 torus 상에 d차원 가상공간을 가지며 이 것은 많은 d차원 영역들로 분할된다. 각 노드들은 하나의 영역에 대응되며 해쉬 함수를 통해 가상공간을 분할한영역의 범위에 해당하는 데이터를 저장한다. 두 개 노드가 속한 좌표 값 중 d-1차원의 값이 중첩되며 한 개 차원이 이웃하면 이웃노드라 한다. 각 노드들은 이들 이웃 노드들의 영역 정보와 IP주소를 가진 라우팅 테이블을 갖는다.

3. 이동 객체 궤적 분산 색인 방법

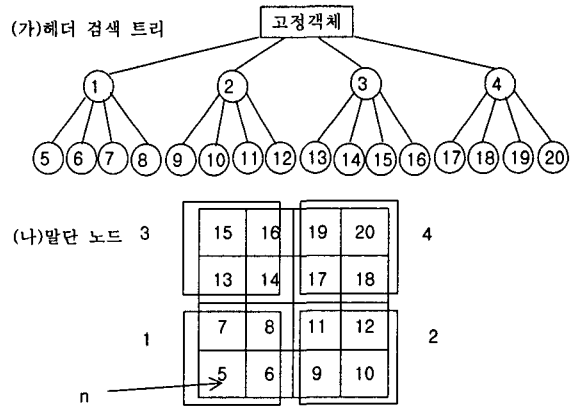
각각의 이동 객체가 자신의 궤적을 관리하는 시스템에서 가장 중점적으로 고려해야 할 것 중의 하나는 영역 질의나 최근접 질의 등을 처리할 때 어떻게 해당 객체를 효율적으로 찾을 수 있는가 이다. 특정 시간에 어떤 지역에 있었던 객체들을 알기 위해서는 질의 처리를 위한 인덱스가 있어야 하고, 중앙 서버가 없는 환경을 고려한다면 그것은 네트워크 상에서 분산되어야 할 것이다. 이에 본 논문에서는 쿼드 트리를 오버레이 네트워크에서 구현하여 영역 질의나 최근접 질의 등을 처리할 수 있게 한다.

본 논문에서 제시하는 이동 객체 과거 궤적 관리 모델의 기본 개념은 각각의 이동 객체는 GPS를 통해 자신의 궤적 데이터를 관리하고, 이동 객체가 활동하는 전체 공간을 2^n*2^n으로 그리드화 하여, 그리드의 각 셀별로 질의 처리를 위한 엔트리 포인터를 제공하는 것이다. 이때, 전체 공간을 2^n*2^n으로 나누는 이유는 새로운 객체가 자신이 위치한 셀의 헤더 객체를 찾을 때 이용하는 헤더 검색 트리를 쿼드 트리[4]로 만들기 위해서이다.

<표 1. 용어정리>

이름	의미
전체 영역	이동 객체들이 활동하는 전체 공간. 열려있는 공간이기 때문에 새로운 객체가 들어오거나, 기존의 객체가 나갈 밖으로 나갈 수 있다.
헤더 객체	각각의 셀은 객체가 자신의 영역에 들어오고 나가는 기록을 관리하는 객체. 질의 처리의 엔트리 포인터
고정 객체	전체 시스템에서 하나가 존재. 맨 처음 객체가 셀의 헤더 객체를 찾으려고 할 때, 최초 접속하는 객체. 이것의 IP는 모든 이동 객체들이 알고 있다고 가정한다. 전체 영역내의 객체의 총수를 통해 헤더검색트리가 완성되어 있는지를 알 수 있다.
새로운 객체	전체 영역 밖에서 안으로 새로 진입한 객체
헤더검색트리	새로운 객체가 현재 위치한 셀에서 그 셀의 헤더 객체를 찾으려고 할 때, 이용하는 쿼드트리
입출입리스트	객체가 특정 셀에 들어온 사건과 특정 셀을 나간 사건

전체 영역을 4*4로 나눈 공간을 예로 들었을 때 헤더 검색 트리의 모양은 <그림 1>의 가)와 같다.



<그림 1. 헤더 검색 트리 구조와 말단 노드>

<그림 1> 의 (가)에서 말단 노드들은 (나)에서처럼 각각 셀의 헤더 객체가 된다. 그러나 셀의 헤더 객체가 된다는 것이 공간적으로 그 셀에 있다는 것을 뜻하지는 않는다. 모바일 IPv6의 무선IP가 공간적인 제약성을 벗어날 수 있는 성질에 의해 헤더 객체는 어느 위치에서든 자신이 담당하는 셀의 입출입 리스트를 관리할 수 있다.

(나)에서 n이 헤더 객체 5가 담당하는 셀에 처음으로 진입하면, n은 최초 고정 객체에 접속하여 자신의 좌표에 의해 객체 1에 접속하게 된다. 객체 1의 자식 노드 중 위치 값에 의해 최종적으로 헤더

객체 5에 자신의 진입을 등록하게 된다.

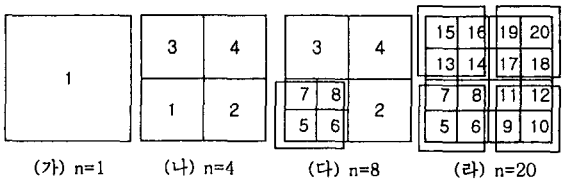
3.1. 헤드 검색트리 초기화 알고리즘

전체 영역에 이동 객체가 새롭게 진입해 들어오면 먼저 자신이 위치한 셀의 헤더 객체를 검색한다. 헤더 객체의 검색을 위해서 본 논문에서는 하나의 고정 객체가 존재하며, 이 고정 객체의 IP는 모든 이동 객체들이 알고 있다고 가정 한다.

객체 등록 알고리즘

```
step1
Access stationary object
step2
IF IsTreeComplete() {
        SearchHeadObject();
        RegisterPosition();
}
ELSE {
        IF total count(object) % 4 == 0 {
                DevideCell();
                RemoveCandidate();
        }
        ELSE {
                RegisterCandidate();
        }
        Search head object
        RegisterPosition();
}
```

트리에서 각 레벨의 노드들은 이웃한 노드들의 정보를 관리한다. 따라서 한번 헤더 객체에 등록한 객체가 다른 셀로 이동하게 되면 헤더 탐색 트리를 이용하지 않고 이전 셀의 헤더 객체가 가지고 있던 이웃 헤더 객체 정보를 통해 바로 등록할 수 있다.



<그림 2. 새로운 객체 등록>

[예제] <그림 2>와 같이 전체 영역이 4*4의 16개의 셀로 이루어져 있다고 하자. (가)에서 전체 영역에 이동 객체가 하나뿐이라면, 그 한 개의 객체 즉 객체 1은 전체 16개 셀의 헤더 객체가 된다. 그래서 자신의 궤적 데이터를 관리하는 것은 물론, 셀과 셀 사이를 이동할 때의 입출입 리스트 16개 모두를 관리하여야 한다. 이러한 상황은 전체 이동 객체가 3개로 늘어나도 아무런 변화가 없다. 객체 2, 3이 새롭게 전체 영역에 들어와도 그들은 자신이 위치한 셀의 헤더 객체(모두 객체 1이지만)에 자신의 존재를 등록할 뿐이다. 그러나 (나)처럼 전체 객체의 수가 4개가 되면, 전체 영역을 사분하여 관리하게 된다. 그래서 처음에 혼자서 모든 셀을 관리하던 객체 1은 나눠진 영역의 셀 입출입 리스트를 해당 헤더 객체 2, 3, 4에게 넘겨주게 된다.

이어서 (다)와 (라)처럼 전체 객체의 개수가 4의 배수가 될 때마다, 영역을 사분하게 된다. 그러다가 트리의 말단 노드가 전체 셀의 수만큼 나누어지면, 더 이상의 영역 분할은 없게 된다. 따라서 그 다음 번의 새로운 객체의 진입 시에는 헤더 객체에 등록하는 과정만 이루어지게 된다.

3.2. 객체의 이동에 의한 궤적 정보의 갱신

전체 공간 내에서 객체의 이동은 <그림 3>과 같이 셀 내의 이동, 셀 사이의 이동, 전체 영역에 입출입 등의 세 가지로 나뉠 수 있다.

- **셀 내에서의 이동**

(가)는 셀 내의 이동은 이동 객체 자신의 궤적 정보만이 변경될 뿐 헤더 객체에 보고하지는 않는다.

- **다른 셀로의 이동**

(나)의 셀 간의 이동에는 객체 자신의 궤적 정보뿐 아니라, 헤더 객체의 입출입 리스트에 새로운 데이터가 추가되게 된다. (나)에서 m이라는 객체가 14:20 이라는 시간에 4가 헤더 객체인 셀에서 2가 헤더 객체인 셀로 이동했다면, 다음과 같은 일들이 내부적으로 일어나게 된다.

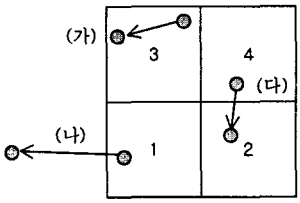
헤더 객체 4의 입출입 리스트에 'IP m을 가지 객체가 14:20에 셀을 나갔다'라는 *[14:20/m/out]* 기록이 더해지고, 헤더 객체 2의 입출입 리스트에 'm이 14:20분에 셀에 들어왔다'라는 *[14:20/m/in]* 기록이 더해진다.

- **전체 영역 밖으로의 이동**

마지막으로 전체 영역에 새로운 객체가 들어오거나 빠져나가는 (다)의 경우에는 상황에 따라 헤더 객체 검색 트리에도 변화를 줄 수 있다. 새로운 객체가 들어오는 상황은 앞 장에 설명이 있었으므로 생략하겠다.

객체가 전체 영역을 나가는 경우에 만약 그 객체가 헤더 검색 트리를 구성하는 객체가 아니라면, 단순히 (다)에서 헤더 객체 1의 입출입 리스트에 객체가 나갔다라는 정보만 추가하면 된다. 그러나 만약 헤더 검색 트리를 구성하던 객체라면 그 객체가 관리하던 정보를 후보 객체에게 넘겨주어야 한다. 객체가 헤더 검색 트리에서 중간 노드라면, 자신의 이웃 노드와 부모 노드, 자식 노드의 정보가 넘겨져야

한다. 객체가 헤더 검색 트리에서 말단 노드 즉, 헤더 객체라면 자신의 이웃 노드와 부모 노드 그리고 입출입 리스트를 넘겨주어야 한다. 정보를 이어 받아 헤더 검색 트리를 유지할 후보 객체는 (다)에서 헤더 객체 1에 등록되어 있는 임의의 객체가 선정된다. 물론, 그 임의의 객체는 헤더 검색 트리의 구성원이 아니어야 한다.



<그림 3. 객체의 이동>

3.3.질의처리

본 논문이 제안하는 모델에서는 영역 질의, 최근접 질의, 과거 궤적 질의를 처리할 수 있다.

● **영역질의**

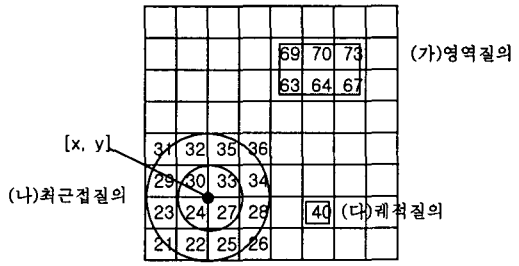
'3시에 질의 영역 q에 있었던 객체의 목록을 구하여라' 와 같은 영역 질의의 경우, <그림 4>의 (가)와 같이 예로 들었을 때 먼저 질의 영역과 겹치는 셀의 헤더 객체를 헤더 검색 트리를 통해 구한다. 6개 셀의 헤더 객체 63, 64, 67, 69, 70, 73의 IP를 얻어 접속한다. 그 다음 헤더 객체의 입출입 리스트에서 질의 시간 조건이 맞는 객체의 IP를 얻어 접속하여 질의 시간과 영역을 검사하여 결과를 반환한다.

● **최근접질의**

'3시에 좌표 (x, y)를 중심으로 가장 가까이 있었던 k개의 객체를 구하여라'와 같은 최근접 질의의 경우, <그림 4>의 (나)와 같이 [x, y]를 접하고 있는 셀의 헤더 객체 24, 27, 30, 35의 IP를 헤더 검색 트리를 통해 얻어 접속한다. 입출입 리스트에서 질의 시간에 그 셀에 있었던 객체의 IP를 얻어 접속한다. 그리고 그 객체의 궤적 정보에서 질의의 [x, y]좌표와 [x, y]에 접하지 않는 셀들(21, 22, 23, 25, 26, 28, 29, 31, 32, 34, 35, 36)과의 최소 거리를 반경으로 하는 (나)의 작은 원에 포함되는 것만 추려서 질의를 수행한다. 왜냐하면 셀 24, 27, 30, 35 밖에 있지만 그 셀들 안에 있는 것보다 [x, y]에 가까운 객체들이 있을 수 있게 때문이다. 만약 해당되는 것이 없다면, 같은 방식으로 영역을 확장하여 (나)의 큰 원을 범위로 수행한다.

● **궤적질의**

'3시에 질의 영역 q에 있었던 객체의 3시부터 4시 사이의 궤적을 구하여라'라는 궤적 질의 경우, <그림 4>의 (다)에서 질의 궤적 영역에 오버랩 되는 셀의 헤더 객체 40에 접속하여 입출입 리스트로부터 후보객체를 얻어온다. 그리고 그 후보 객체에 접속하여 질의 조건에 부합하는 지를 검사한다. 만약 조건 부합하면 3시부터 4시 사이의 궤적을 반환한다.



<그림 4. 질의의 종류>

4.결론 및 향후연구

본 논문에서 제안하는 모델은 질의 처리와 궤적 정보의 갱신, 그리고 통신 트래픽 분산을 위하여 이동 객체들이 자기 자신의 궤적을 관리함으로써 수십, 수백만 개의 객체가 있는 환경에서 효율적으로 관리가 가능하다. 또한 이동 객체의 위치 변화에 따른 갱신의 횟수가 다른 셀로의 이동에 한정됨으로써 중앙 서버가 관리하는 경우와 비교하여 상당히 감소함을 알 수 있다.

향후 연구로는 공간을 도로네트워크의 세그먼트의 집합으로 분할하여 구현해볼 것이다.

참고문헌

[1] F. Harrell, Y. Hu, G. Wang, and H. Xia, "Survey of locating and routing in peer-to-peer systems," http://www.sics.se/~sameh/research/P2P/Surveys and Comparisons/Surveyof Locating and Routing in p2p systems/group15.pdf.

[2] I. Stoica, R. Morris, D. Karger, F. Kaashoek and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for Internet application," in Proc. ACM SIGCOMM 2001.

[3] S. Ratnasamy, P.Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in Proc. of ACM SIGCOMM, 2001.

[4] http://www.hit.co.kr/homepage/file/itreport/HIT- itreport-0242992747-1007.pdf