분산 그리드 환경에서 힐버트 커브를 이용한 효율적인 Cloaking 영역 설정 기법

(A Efficient Cloaking Region Creation Scheme using Hilbert Curves in Distributed Grid Environment)

이 아 름* 엄 정 호** 장 재 우***
(Ah Reum Lee) (Jung Ho Um) (Jae Woo Chang)

요 약 최근 무선 통신과 모바일 측위 기술의 발전으로 위치 기반 서비스(Location-Based Service)의 이용이 확산되었다. 그러나 위치 기반 서비스에서 사용자는 사용자의 정확한 위치를 가지고 데이터베이스 서버에 질의를 요청하기 때문에, 사용자의 위치 정보가 상대방에게 노출될 수 있다. 따라서 모바일 사용자에 의한 안전한 위치기반 서비스의 사용을 위해서는 사용자의 개인 정보 보호 방법이 요구된다. 이를 위해 본 논문에서는 위치기반 서비스에서 사용자의 위치정보를 보호하기 위하여, 분산 그리드 환경에서 힐버트 커브를 이용한 효율적인 cloaking 영역 설정 기법을 제안한다. 제안하는 기법은 K-anonymity를 만족하는 cloaking 영역을 생성하기 위해 힐버트 커브의 특성을 분석하고 이를 통해 이웃 셀의 힐버트 커브값을 계산하여 최소화된 cloaking 영역을 설정한다. 아울러, 네트워크 통신비용을 줄이기 위해 분산 해쉬 테이블 구조인 Chord를 사용한다. 마지막으로 성능평가를 통해서 제안하는 기법이 기존의 그리드 기반 cloaking 기법보다 우수함을 보인다.

키워드 : 개인 정보 보호, 위치기반 서비스, 분산 그리드 환경, Cloaking 기법, Chord

Abstract Recent development in wireless communication and mobile positioning technologies makes Location-Based Services (LBSs) popular. However, because, in the LBSs, users request a query to database servers by using their exact locations, the location information of the users can be misused by adversaries. Therefore, a mechanism for users' privacy protection is required for the safe use of LBSs by mobile users. For this, we, in this paper, propose a efficient cloaking region creation scheme using Hilbert curves in distributed grid environment, so as to protect users' privacy in LBSs. The proposed scheme generates a minimum cloaking region by analyzing the characteristic of a Hilbert curve and computing the Hilbert curve values of neighboring cells based on it, so that we may create a cloaking region to satisfy K-anonymity. In addition, to reduce network communication cost, we make use of a distributed hash table structure, called Chord. Finally, we show from our performance analysis that the proposed scheme outperforms the existing grid-based cloaking method.

Keywords : Privacy Protection, Location-Based Services, Distributed Grid Environment, Cloaking Scheme, Chord

1. 서 론

최근 무선 통신과 모바일 위치 측정 기술의 발전으로 위치기반 서비스(Location-Based Service)의 이용이 확산되었다. 위치기반 서비스란 유무선 통신망을 통해 얻은 위치정보를 다른 유용한 정보와 실시간으로 결합하여 사용자가 필요로 하는 부가적인 응용 서비스를 제공하는 것으로 정의된다[1,2]. 위치기반 서비스에서 모바일 사용자가 GPS 등과 같은 위치 측정 장치를 이용하여, 사용자의 위치 정보를 데이터베이스 서버에 보내어 교통 정보, 친구 찾기, 인접한 POI(Point Of Interest)등 다양한 종류의 위치 기반 서비스를 이용할 수 있다. 그러나 이와

† 이 논문은 2009년도 정부(교육과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구임(No. 2009-0059417)

* 전북대학교 전자정보공학부 컴퓨터공학과 석사과정, arlee@dblab.chonbuk.ac.kr
** 전북대학교 전자정보공학부 컴퓨터공학과 박사과정, jhum@dblab.chonbuk.ac.kr
*** 전북대학교 전자정보공학부 컴퓨터공학과 교수, jwchang@chonbuk.ac.kr(교신저자)

논문접수 : 2009.01.15 수정일 : 1차 2009.03.03 / 2차 2009.03.19 심사완료 : 2009.03.20

같이 사용자의 정확한 위치정보를 가지고 데이터베이스 서버에 서비스를 요청하는 것은 심각한 개인 정보 누출의 위협이 될 수 있다. 상대방(adversary)이 서비스 이용자들이 어떤 장소에 자주 방문하는지, 또한 이러한 방문이 어떤 시간대에 주로 이루어지는 지를 파악하여, 생활 스타일, 질병 정보 등의 사생활 정보를 획득할 수 있기 때문이다. 실제로 국외의 경우 위치 기반 서비스를 이용한 스토킹 피해 사례가 빈번히 발생하고 있다[3,4]. 따라서 모바일 사용자의 안전하고 편리한 위치기반 서비스 사용을 위한 개인 정보 보호 방법이 요구된다.

위치기반 서비스에서 개인 정보 보호를 위한 연구 중에는 K-anonymity를 만족하는 cloaking 기법을 활용한 연구가 존재한다. 이 기법은, 사용자가 데이터베이스 서버에 질의(서비스) 전송 시, 사용자의 좌표정보를 숨기기 위해 K-anonymity를 만족하면서 최소 크기를 가지는 질의 영역(이하 cloaking 영역)을 설정하는 것을 말한다. 여기서 K-anonymity는 cloaking 영역에 질의를 요청한 사용자와 그 사용자를 제외한 K-1명의 다른 사용자의 위치 정보를 포함하는 것이다. 이러한 cloaking 영역의 설정으로 사용자의 위치 노출 확률을 최소화 시킬 수 있다. 이러한 K-anonymity를 고려한 cloaking 기법들은 cloaking 영역을 설정하는 주체에 따라 중앙 집중(Centralized) 방식과 분산(Decentralized) 방식으로 분류한다. 중앙 집중 방식은 모바일 사용자와 데이터베이스 서버 중간에 신뢰할 수 있는 서버(이하 anonymizer)를 존재시켜, anonymizer가 질의에 포함된 사용자의 위치정보를 보호하는 cloaking 영역을 설정하는 방법이다. 중앙 집중 방식은 cloaking 영역을 설정하는 시간은 매우 빠르지만, anonymizer에서의 병목(bottleneck) 현상으로 인한 성능 저하 문제 및 사용자의 위치 정보를 저장하고 있는 anonymizer의 보안 위협 문제가 존재한다. 이와 같은 문제점을 해결하기 위해 분산 방식은 cloaking 영역 설정 시 중앙의 anonymizer를 사용하지 않고, cloaking 영역을 요구하는 사용자가 다른 사용자들과의 통신을 통하여 직접 cloaking 영역을 설정한다. 분산 방식은 사용자들 간의 통신비용으로 cloaking 영역 설정 속도는 중앙 집중 방식보다 느리지만, 분산 구조로 인한 높은 신뢰성을 제공한다.

분산 방식의 기존 cloaking 기법들 중 가장 우수한 성능을 보이는 연구는 G. Ghinita et al.이 제안한 MOBIHIDE[5]이다. MOBIHIDE는 힐버트 커브를 이용하여 사용자의 위치를 1차원으로 변환한 후, 사용자들의 위치를 기반으로 cloaking 영역을 설정한다. 그러나 이 연구는 두 가지의 문제점을 지닌다. 첫째, 사용자가 요구하는 K-anonymity를 만족할 때까지 힐버트 커브를 따라 사용자를 탐색하기 때문에 cloaking 영역의 크기가 증가한다. 둘째, 사용자를 탐색하기 위해 이웃 클러스터 노드를 순차검색하기 때문에 네트워크상의 통신비용을 증가시킨다. 따라서 본 논문에서는 분산 그리드 환경에서 힐버트 커브를 이용한 효율적인 cloaking 영역 설정 기법을 제안한다. 제안하는 기법은 사용자가 요구한 K-anonymity를 만족하는 cloaking 영역을 설정하기 위해 힐버트 커브의 특성을 분석하고 이를 통해 이웃 셀의 힐버트 커브 값을 계산하여 최소 크기의 cloaking 영역을 설정한다. 아울러, 네트워크 통신비용을 줄이기 위해 Chord[6] 기반의 분산 해쉬 테이블을 개선하여 사용한다.

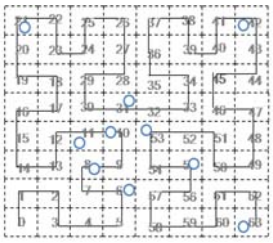
본 논문의 구성은 다음과 같다. 2장에서는 K-anonymity를 고려한 기존 cloaking 기법들을 소개한다. 3장에서는 기존 연구의 문제점을 개선한 힐버트 커브를 이용한 효율적인 cloaking 영역 설정 기법의 구조를 제안하고, 4장에서는 제안한 구조를 바탕으로 cloaking 영역을 설정하는 알고리즘을 제안한다. 5장에서 기존 연구와 본 논문에서 제안한 기법과의 성능 비교를 수행하고, 마지막으로 6장에서는 결론 및 향후 연구에 대해 기술한다.

2. 관련 연구

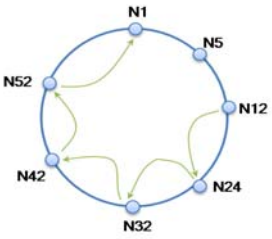
사용자의 위치 정보 보호를 위해 연구된 기존의 cloaking 기법들 중 anonymizer가 cloaking 영역을 설정하는 중앙 집중 방식을 사용한 대표적인 연구는 다음과 같다.

먼저, B. Gedik et al. 연구[7]는 공간 영역 탐색 방법을 사용하여 cloaking 영역을 설정하며, 이 때 서버에 존재하는 사용자들의 이웃(neighbor)관계를 동적 그래프로 유지하는 clique를 사용한다. 또한 M. F. Mokbel et al. 연구[8]는 그리드 기반의 피라미드 데이터 구조를 사용하여 cloaking 영역을 설정하며, 사용자의 밀집도가 높은 장소에서 cloaking 영역이 작게 설정되는 것을 막기 위하여 최소 cloaking 영역 크기를 설정하는 기법을 제안한다. 아울러, J. Chang et al. 연구[9]는 R*-tree기반의 색인 구조를 사용하여 사용자가 요구한 L-diversity를 수행한 뒤, K-anonymity를 수행하여 최소의 cloaking 영역을 설정하는 기법을 제안한다. 여기서 L-diversity란, K-anonymity와 보완적으로 사용되는 것으로, cloaking 영역 내에 L개의 다른 장소(우편물이 배달될 수 있는 주소지 기준)를 포함하는 것을 말한다. 그러나 중앙 집중 방식에서의 cloaking 기법들은 cloaking 영역을 설정하는 시간은 매우 빠르나, anonymizer에서의 병목(bottleneck) 현상과 사용자의 위치 정보를 모두 저장하고 있는 anonymizer의 보안 위협 문제가 존재한다.

이와 같은 문제점을 해결하기 위해 분산 방식에서는 cloaking 영역 설정 시 중앙의 anonymizer를 사용하지 않고, cloaking 영역을 요구하는 사용자가 다른 사용자들과의 통신을 통하여 직접 cloaking 영역을 설정한다. 분산 방식을 이용한 cloaking 기법에 대한 연구로는 세 가지가 존재한다. 먼저, C. Y. Chow et al. 연구[10]는 질의를 요청한 모바일 사용자가 자신의 통신 범위 내에 있는 K-1명의 다른 사용자를 찾아 cloaking 영역을 설정하는 기법을 제안하였다. 이 연구는 기존 연구들이 중앙 집중 방식을 제안했던 반면, 처음으로 분산 방식을 제안된 것에 의의가 있다. 하지만, 설정된 cloaking 영역의 중심에 질의를 요청한 사용자가 위치할 확률이 높은 문제점을 지닌다. 두 번째 연구는 G. Ghinta et al.이 제안한 PRIVE[11]로, 힐버트 커브를 이용하여 C. Y. Chow et al. 연구의 문제점을 해결하였다. 이 연구는 B+-트리와 유사한 분산된 색인 구조를 사용하여 cloaking 영역을 설



(a) 그리드상의 사용자 분포



(b)클러스터 대표노드 구성

구성원	C6, C8, C10, C11		
선행자	N5		
후행자	N24		
finger	Entry	Id	대표노드
	$N12+1(2^0)$	13	N24
	$N12+2(2^1)$	14	N24
	$N12+4(2^2)$	16	N24
	$N12+8(2^3)$	20	N24
	$N12+16(2^4)$	28	N32
	$N12+32(2^5)$	44	N52

(c) 대표노드 저장구조

그림 1. MOBIHIDE에서 사용된 Chord 구조

정하지만, 계층적 색인 구조의 사용으로 인해 부하 분산에 취약점을 갖는다. 마지막으로, G. Ghinta et al.이 제안한 분산 방식의 cloaking 기법 중 가장 우수한 성능을 보이는 MOBIHIDE[5]가 존재한다. MOBIHIDE는 PRIVE의 부하 분산 문제를 해결하기 위해, Chord 기반의 분산 해쉬 테이블 구조를 사용한다. 이 기법은 힐버트 커브를 따라 사용자를 탐색하고, 랜덤한 식별자를 고려하여 cloaking 영역을 설정하기 때문에 기존의 방법에 비해 높은 신뢰성을 갖는다. MOBIHIDE에서 사용하는 Chord는 링 형태의 분산 해쉬 테이블을 갖는 구조방식의 p2p 프로토콜로써, m-bit(0~2^m-1)의 원형 주소 공간에 각각의 노드와 데이터의 키 값을 할당하는 방법이다. Chord의 각 노드들은 선행노드와 후행노드의 정보를 저장하고, 이웃 노드와의 라우팅을 위하여 finger 테이블을 관리한다. finger 테이블은 주소 공간의 크기에 따라 크기가 달라지는데, m-bit일 경우 m개의 행으로 이루어진다. 각 행에는 현재 노드의 위치를 기반으로 2의 n(1~m) 제곱수에 해당하는 노드의 정보를 저장한다. 위의 방법을 이용하여 MOBIHIDE에서 사용하는 클러스터 구성은 다음과 같다. 먼저, 그림 1-(a)와 같이 분산된 사용자의 위치를 힐버트 커브를 이용하여 힐버트 값으로 변환한다. 이를 기반으로 가상 링 구조를 형성하여 대표 노드를 구성하고, 노드는 링 구조상에서 시계 방향으로 대표 노드의 클러스터에 삽입한다. 각 대표 노드는 그림 1-(c)와 같이 구성원, 선행자, 후행자, finger 테이블이 저장한다. 구성원은 클러스터 내에 포함된 각 노드를 나타내고, 선행자는 선행 클러스터의 대표 노드를 나타내며, 후행자는 후행 클러스터의 대표노드를 나타낸다. finger 테이블은 네트워크상에 노드 검색 시 클러스터 접근을 빠르게 하기 위해, 현재 대표 노드에서 2^1,2^2,...,2^n(n 힐버트 커브 최대값이 가지는 비트 수)의 거리에 있는 노드의 대표 노드를 저장한다.

이를 이용한 MOBIHIDE의 cloaking 영역 설정 기법은 다음과 같다. 먼저, 사용자가 서비스를 요청하면 그리드 상에서 힐버트 커브를 생성하여 사용자의 위치에 해당하는 힐버트 커브 값에 식별자를 부여한다. 한편, cloaking 영역 설정을 위해 힐버트 커브 값을 임의로 증가 또는 감소(이하 H) 시킨다. 만약, H가 최대 힐버트 커브 값을 초과하면 H에서 최대 힐버트 커브 값을 감산한다. 무선 네트워크에 존재하는 이웃한 사용자를 탐색하기 위해 H셀에 해당하는 클러스터를 현재 클러스터의 대표 노드에 저장된 라우팅 정보를 통해 접근한다. 해당 클러스터를 탐색하여 H셀의 사용자의 수를 획득하고, 이를 현재까지 탐색한 셀들의 모든 사용자 수(이하 M)와 합산한다. 만약 M이 사용자가 요구한 K-anonymity(이하 k)를 만족하지 못하면, H값을 다시 설정한 후 위의 과정을 반복한다. M이 k를 만족한다면, 지금까지 탐색한 셀을 기반으로 cloaking 영역을 설정하여 위치기반 서비스 서버에 전송한다. 예를 들어, 모바일 사용자가 개인정보를 보호하면서 서비스를 이용하기 위해 모바일 사용자를 포함한 k=6인 cloaking 영역을 요청하면, 사용되는 클러스터 구성은 그림 1-(b)같다. cloaking 영역 설정을 요청한 사용자가 그림 2와 같다면, 사용자는 힐버트 커브 값 10을 부여받고, H를 11로 저장한다.

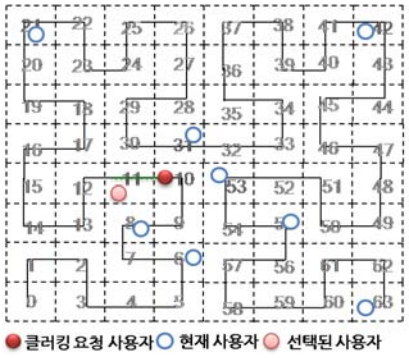


그림 2. 사용자가 서비스를 요청

클러스터는 가상 링 구조상에서 시계 방향으로 구성되기 때문에 힐버트 커브 값 10을 갖는 노드의 대표 노드는 그림 1-(b)에서 N12에 해당한다. N12에서 H에 해당하는 셀이 존재하므로, 해당 셀에 접근하여 사용자의 수를 획득하고, M을 구한다. 이와 같이 힐버트 커브 값을 임의로 증가시켜 H값을 설정하고 H에 해당하는 셀을 N12를 기준으로 탐색한다. 만일 H에 해당하는 셀이 N12에 존재하면 셀에 접근하여 사용자 수를 획득한다. 하지만, 해당 노드의 클러스터에 셀이 존재하지 않으면 대표

노드가 저장하고 있는 선행자, 후행자 또는 finger 테이블을 이용하여 H에 해당하는 셀을 탐색한다. H에 해당하는 셀에 사용자가 없다면, H값을 다시 설정한 후 위의 과정을 반복한다. 위의 과정을 통해 결과적으로 N24, N32, N42, N52, N1의 노드를 순차적으로 방문한다. 이는 k를 만족할 때까지 사용자를 탐색하기 때문에, 그림 3과 같이 k를 만족하는 사용자가 셀 11, 21, 31, 42, 53에 존재할 경우, 셀 53의 대표노드인 N1 노드까지 탐색하며, 최종적으로 색칠된 사각형 영역을 cloaking 영역으로 설정한다.

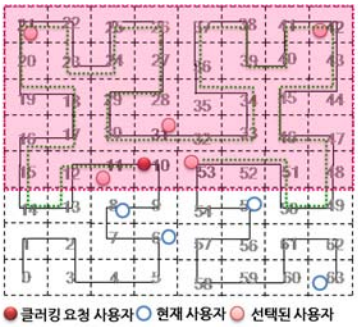


그림 3. MOBIHIDE의 cloaking 영역 설정 결과

3. 힐버트 커브를 이용한 효율적인 Cloaking 영역 설정 기법의 구조

본 장에서는 기존 연구인 MOBIHIDE의 문제점을 기술하고, 그 문제점을 보완하여 분산 그리드 환경에서 힐버트 커브를 이용한 효율적인 cloaking 영역 설정 기법의 구조를 제안한다.

3.1 기존 연구의 문제점

MOBIHIDE의 cloaking 기법은 다음 두 가지 문제점을 지닌다. 첫째, 사용자의 이웃 셀에 사용자가 존재하여도, 사용자가 요청한 K-anonymity를 만족할 때까지 힐버트 커브를 따라 사용자를 탐색하므로 설정되는 cloaking 영역의 크기가 매우 커질 수 있다. cloaking 영역의 크기가 커지면, 위치기반 서비스 서버에 질의 수행 시간이 증가하고, 많은 후보 집합을 포함하기 때문에 질의 처리 결과가 부정확해진다. 둘째, K-anonymity를 만족할 때까지 이웃 클러스터 노드를 후행자 또는 선행자에 따라 순차방문하기 때문에, 최악의 경우 한명의 사용자 위치를 은닉하기 위해 네트워크상의 거의 모든 대표노드를 방문하게 된다. 이는 네트워크에서 통신비용을 증가시킨다. 따라서 본 논문에서는 이와 같은 문제점을 해결하기 위해, 분산 그리드 환경에서 힐버트 커브를 이용한 효율적인 cloaking 기법을 제안한다. 먼저, 제안하는 기법은 힐버트 커브의 특성을 분석하여 이웃 셀의 힐버트 커브 값을 계산하며, 이를 통해 사용자가 요구하는 최소 크기의 cloaking 영역을 설정한다. 아울러, 대표노드에 저장된 finger 테이블을 개선하여 이웃한 사용자를 탐색하기 위한 클러스터의 접근 횟수를 최소로 줄여, 네트워크에서의 통신비용을 감소시킨다.

3.2 cloaking 영역 설정 알고리즘을 위한 시스템 구조

본 논문에서 제안하는 cloaking 기법은 분산 방식을 사용하며, 시스템 구조는 그림 4와 같다.

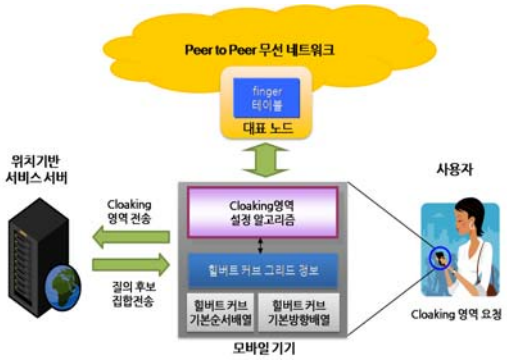


그림 4. 제안하는 기법의 시스템 구조

시스템 구조는 크게 사용자의 모바일기기 및 P2P상에서 모바일기기가 속해 있는 클러스터로 구성된다. 제안하는 기법의 시스템 흐름을 시퀀스 다이어그램으로 나타내면 그림 5와 같다. 사용자의 모바일기기는 힐버트 커브 그리드 정보, 힐버트 커브 기본 순서 배열, 힐버트 커브 기본 방향 배열을 기반으로(그림 4), 사용자가 위치한 이웃 셀을 계산하여 클러스터에 존재하는 이웃셀의 모바일 사용자 정보를 요청한다. 클러스터는 이웃셀의 정보를 기반으로 모바일 사용자들의 정보를 사용자에게 보낸다. 위의 과정을 사용자가 요구하는 K-anonymity를 만족할 때까지 반복하여 최소의 cloaking 영역을 설정한다. 사용자는 설정된 cloaking 영역과 질의를 위치 기반 서버에게 전송하고, 위치기반 서버에서는 질의처리를 한 후, 후보 집합을 사용자에게 전송한다. 사용자는 전송된 후보 집합을 고려하여 정확한 질의 처리를 수행한다.

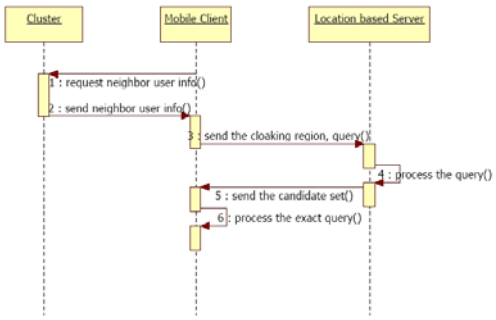
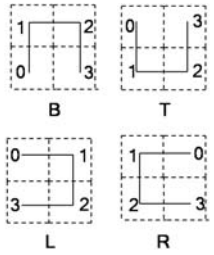


그림 5. 제안하는 기법의 시스템 흐름

제안하는 시스템에서 cloaking 영역 설정 시에 사용되는 세 가지 테이블의 자세한 구성은 다음과 같다. 첫째, 힐버트 커브 그리드 정보 테이블은 힐버트 커브를 통해 생성된 사용자 식별자, 현재 사용자가 위치한 그리드상의 셀 정보, 힐버트 커브의 차수, 힐버트 커브 타입으로 구성된다. 둘째, 힐버트 커브 기본 순서 배열은 2*2 그리드를 구성하는 힐버트 커브의 셀 ID를 저장하고 있다. 2*2 그리드는 힐버트 커브의 최소 단위이며, 힐버트 커브는 시작 위치 및 종료 위치에 따라 4가지 타입의 커브가 생성된다. 각 타입은 그림 6에서처럼 시작 및 종료 방향에 따라 B(그리드 하 방향에서 시작 종료), T(그리드 상 방향에서 시작 종료), L(그리드 좌 방향에서 시작 종료), R(그리드 우 방향에서 시작 종료)로 명명한다. 시작 및 종료 위치가 바뀌는 경우 각 타입에 따라 B-1, T-1, L-1, R-1로 정의한다. 이 타입들의 힐버트 커브 값은 2*2그리드에 할당된 최대 힐버트 커브 값에서 B, T, L, R 타입에 저장된 값을 감산하여 계산한다. 한편, 각 타입에 생성된 힐버트 커브 값을 1차원 배열에 맵핑하기 위하여 식(1)과 같은 맵핑 함수를 사용한다.

$$I = X + 2 * Y \quad (1)$$

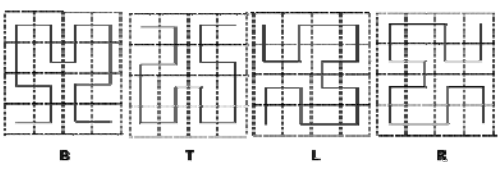
I는 맵핑되는 인덱스를 나타내고, X와 Y는 그리드상의 좌표를 나타낸다. 그림 6은 각 타입의 힐버트 커브를 1차원 배열에 맵핑한 것을 보여준다.

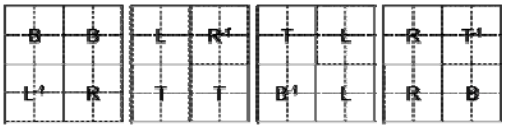


방향 \ 인덱스	0	1	2	3
B	0	3	1	2
T	1	2	0	3
R	2	3	1	0
L	3	2	0	1

그림 6. 힐버트 커브 기본 순서 배열

셋째, 힐버트 커브 기본 방향 배열은 2*2 그리드를 구성하는 힐버트 커브의 각 타입을 저장하고 있다. 그림 7은 4*4 그리드에서 힐버트 커브의 각 타입에 따라 커브가 생성된 모습을 나타낸다. 이를 살펴보면, 4*4 그리드의 커브는 4개의 2*2 그리드의 힐버트 커브로 구성된다. 각 4*4 그리드 힐버트 커브에 대하여, 이를 구성하고 있는 2*2 그리드 힐버트 커브 타입을 표시하면 그림 8과 같다. 이는 일반적인 정방향 그리드에서, 그리드를 4등분하여 각 그리드에 대한 타입을 나타낸다. 힐버트 커브 기본 방향 배열은 B, T, L, R 타입만 저장하며, B^{-1}, T^{-1}, L^{-1}, R^{-1} 타입은 저장된 타입 구성을 역변환하여 계산한다. 즉, 예를 들어 B={B, B, L^{-1}, R}이므로 B^{-1}은 B^{-1}={B^{-1}, B^{-1}, L, R^{-1}}이 된다. 그림 8은 힐버트 커브 기본 방향 배열을 1차원 배열에 맵핑한 것을 보여준다.





방향 \ 인덱스	0	1	2	3
B	L^{-1}	R	B	B
T	T	T	L	R^{-1}
R	R	B	R	T^{-1}
L	B^{-1}	L	T	L

그림 8. 힐버트 커브 기본 방향 배열

모바일기기가 위치해 있는 클러스터의 대표 노드는 이웃한 사용자를 탐색하기 위한 클러스터로의 접근 횟수를 최소화하기 위해 개선된 finger 테이블을 사용한다. 이를 사용함으로써 제안하는 cloaking 영역 설정 기법은 네트워크 통신비용을 줄이면서 최소의 cloaking 영역을 설정할 수 있다. 그림 9는 클러스터의 대표노드에 저장된 finger 테이블을 나타낸다. finger 테이블은 클러스터 구성원 정보, 선행자, 후행자 및 현재 대표 노드에서 전후로 2^0,2^1,...,2^{N-1}(N은 그리드의 차수 8*8 그리드는 N=6)의 거리에 있는 셀이 포함된 클러스터 대표 노드를 저장한다. 구성원 정보는 사용자가 있는 셀 ID와 셀에 존재하는 사용자 수(m)를 저장한다. 선행자는 선행 클러스터의 대표 노드를 저장하고, 후행자는 후행 클러스터의 대표노드를 저장한다. finger 테이블은 클러스터 대표 노드까지 빠르게 접근하기 위해 사용하고, 셀과 이웃 셀의 차이가 전체 그리드 크기의 1/2이상 커지지 않기 때문에 $2^3*2^3=2^6/2=2^5$까지의 거리에 있는 셀의 대표노드를 저장한다.

구성원	$C_1(m), C_2(m), \ldots, C_n(m)$			
선행자	N_{i-1}			
후행자	N_{i+1}			
finger	Entry	후행자	Entry	선행자
	$N_i+1(2^0)$	N_j	$N_i-1(2^0)$	N_L
	$N_i+2(2^1)$	N_{j+1}	$N_i-2(2^1)$	N_{L+1}
	$N_i+4(2^2)$	N_{j+2}	$N_i-4(2^2)$	N_{L+2}
	...	...	...	....
	N_i+2^{N-2}	N_{j+N-2}	N_i-2^{N-2}	N_{L+N-2}
	N_i+2^{N-1}	N_{j+N-1}	N_i-2^{N-1}	N_{L+N-1}

그림 9. 클러스터 대표노드의 finger 테이블

그림 7. 4*4 그리드에서 4가지 힐버트 커브 타입

4. 힐버트 커브를 이용한 효율적인 Cloaking 영역 설정 알고리즘

본 장에서는 3장에서 제안한 시스템 구조를 바탕으로 힐버트 커브를 이용한 효율적인 cloaking 영역 설정 알고리즘을 제안한다.

4.1 힐버트 커브를 이용한 효율적인 cloaking 영역 설정 알고리즘

cloaking 영역 설정 알고리즘을 제안하기 위해, 먼저 힐버트 커브 값과 좌표간의 관계를 분석한다. 예를 들어, 8*8 그리드에서 (4,3) 의 좌표에 해당하는 힐버트 커브 값은 53이다(그림 10-(a)). 그리드를 분할해가면서 힐버트 커브 값을 분석해보면, 먼저 8*8 그리드를 4등분하면 힐버트 커브 값 53은 4사분면에 있다. 4사분면은 각 사분면을 좌표로 표현했을 때, (1,0)에 해당한다. 다시, 4사분면의 그리드 즉, 4*4 그리드에서 이를 4등분하면 힐버트 커브 53은 1사분면에 있고, 이를 다시 4등분하면 그리드의 1사분면에 있다. 1사분면은 좌표로 표현하면 (0,1)에 해당한다. 즉, 힐버트 커브 값 53을 사분면으로 분할하였을 때 각 분할 지점에서 좌표는 8*8 그리드에서 (1,0), 4*4 그리드에서 (0,1), 2*2 그리드에서 (0,1) 이다. 이는 (4,3)을 비트로 표현한 (100,011) 좌표를 비트별로 분할한 것과 동일하다. 8*8의 서브 그리드는 4*4 그리드이고 크기는 16이다. 이를 토대로 53을 계산하면 48+4+1=16*3+4*1+1*1이 된다. 이는 8*8 그리드에서 2*2 그리드로 변환할 때 (1,0)에 해당하는 힐버트 커브 값이 3임을 의미한다. 이는 힐버트 커브 타입 중에서 B타입에 속하며, 각각을 계산하면 다음과 같다.

8*8 그리드(1,0) → 3, B
4*4 그리드(0,1) → 1, R
2*2 그리드(0,1) → 1, R

즉, 좌표를 비트별로 분해하고 이를 힐버트의 기본 타입에 대입한 후 해당하는 서브 그리드의 크기를 곱하여 더하면, 힐버트 커브 값을 좌표를 통해 직접 계산할 수 있다.

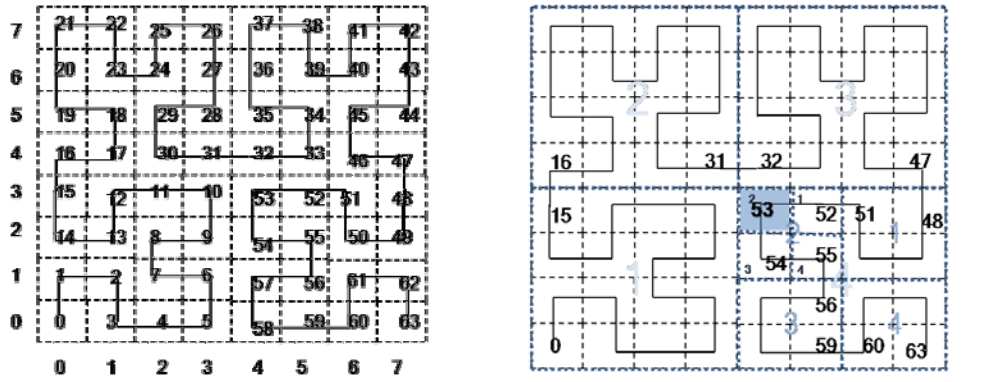
본 논문에서 제안하는 cloaking 영역 설정 알고리즘은 크게 사용자 셀의 이웃 셀을 계산하는 부분과 이웃 셀에 있는 사용자를 탐색하는 부분으로 구성된다. 먼저, 수행단계 1은 사용자의 K-anonymity를 만족하는 cloaking 영역을 설정하기 위해, 사용자 셀의 이웃 셀을 계산하는 방법을 기술한다.

수행단계 1. 사용자 셀의 이웃 셀을 계산

모바일 기기의 힐버트 커브 그리드 정보 테이블에서 힐버트 커브 방향 D와 힐버트 커브 차수 G를 입력 받고, 현재 셀의 좌표를 이용하여 이웃 셀의 좌표를 계산한다. 이웃 셀 좌표가 계산되면, 이를 비트로 표현하고 반복 횟수를 카운트(count)하는 변수 i를 초기화한다. 좌표를 비트로 표현하면 힐버트 커브 차수 i 비트만큼 이동(shift)하여 1과 AND 비트 연산을 취하고, 식(1)의 맵핑 함수에 따라 1차 배열에서의 인덱스 값(Ri)을 계산한다. 이를 표현하면 식 (2)와 같다.

$$R_i = (X \gg ((G-1)-i)\&1) + 2^*(Y \gg (G-1)-i)\&1) \quad 0 \le i \le G-1 \tag{2}$$

인덱스 값이 계산되면 모바일기기에 저장된 힐버트 커브 기본 순서 배열 중에서 힐버트 커브 방향 D에 해당하는 타입에 따라 비트 연산된 좌표에 해당하는 값을 V에 저장한다. 아울러 힐버트 커브 기본 방향 배열에서 비트 연산된 좌표에 해당하는 값을 다시 힐버트 커브 방향 D에 저장한 후, G를 감소시켜 위의 작업을 반복한다. 이 작업은 G가 0보다 작아질 때까지 반복한다. 힐버트 커브 기본 순서 배열의 값을 얻어오는 함수를 OHilb(Row, Column), 힐버트 커브 기본 방향 배열의 값을 얻어오는 함수를 DHilb(Row, Column)라 하면, 이웃 셀 C의 값은 각 과정에서 계산된 V의 값을 각 반복 단계에서 서브 그



(a) 8*8그리드의 힐버트 커브 생성 모습 (b)8*8그리드에서 각 서브 그리드의 시작 및 종료 힐버트 커브 값

그림 10. 힐버트 커브 값과 좌표간의 관계

리드 크기와 곱하여 이를 모두 합산한 값이다. 이에 대한 계산 과정은 식(3)~식(5)와 같다.

$$V_i = OHilb(R_i, D_i) \tag{3}$$

$$D_i = \begin{cases} i = 0 \;\; D \\ i \neq 0 \;\; DHilb(R_{i-1}, D_{i-1}) \end{cases} \tag{4}$$

$$C = \sum_{i=0}^{2} (V_i * 2^{2*(G-1-i)}) \tag{5}$$

이웃 셀을 방향 별로 계산하였다면, 수행단계 2에서는 cloaking 영역을 설정하기 위해 k를 만족할 때까지, 이웃 셀에 있는 사용자를 탐색한다.

수행단계 2. 이웃 셀에 있는 사용자를 탐색

클러스터를 최소로 방문하면서 이웃한 사용자를 찾기 위해, 이웃 셀 탐색 순서는 식 (6)과 같이 현재 셀과 가장 가까운 힐버트 커브 값의 순서로 계산하며, 만일 최소 값이 하나 이상일 경우에는 힐버트 커브 값이 작은 것부터 탐색한다.

$$C_{\min} = Min((H - C_1),(H - C_2),(H - C_3),(H - C_4)) \tag{6}$$

탐색할 셀이 결정되면 현재 사용자가 속해 있는 대표 노드의 finger 테이블을 검색하여, 탐색할 셀에 해당하는 클러스터를 결정한다. 클러스터 결정은 클러스터 대표 노드의 힐버트 커브 값에서 탐색할 셀의 값을 감산하여 그 값이 가장 작은 값을 선택한다. 단, 감산 시 음수로 계산될 경우, 힐버트 커브의 최대값을 더하여 보정한다. 이는 클러스터의 대표노드가 링 구조로 힐버트 커브 값이 순환하기 때문이다. 클러스터 대표 노드를 N이라 할 때 클러스터 결정 식은 식(7), 식(8)과 같다. 단, n은 finger 테이블의 클러스터 대표 노드의 개수이다.

$$N_{diff}(i) = N - C_i \tag{7}$$

$$\text{if } N_{diff}(i) < 0 \; then \; N_{diff}(i) = N_{diff}(i) + 2^{2G} - 1$$
$$N_{\min} = \min(N_{diff}(0), N_{diff}(1), ..., N_{diff}(n)) \tag{8}$$

접근해야 할 클러스터가 결정되면 해당 클러스터에 접근하여 탐색할 셀의 사용자 수를 검색하여 합산한다. 합산한 사용자 수가 사용자가 요청한 k보다 작다면 다음 이웃 셀을 탐색한다. 만일 탐색할 이웃 셀이 없다면, 확장할 기준 셀을 가장 먼저 탐색한 이웃 셀로 설정하고 이웃 셀 좌표를 계산한다. 만일 k를 만족한다면, 검색한 사용자를 포함하는 cloaking 영역을 설정하고 이를 위치기반 서비스 서버에 전송한다. 앞에서의 내용을 바탕으로 설계한 인접셀을 계산하는 cloaking 영역 설정 알고리즘은 그림 11과 같다. 먼저 cloaking 영역을 요청하는 사용자의 위치정보, cloaking 영역 내에 포함되길 원하는 사용자 수 K, 힐버트 커브 방향 D 와 힐버트 커브 차수 G를 입력받는다. 다음으로 사용자의 위치정보를 기준으로 이웃셀의 좌표를 계산한다. 이웃셀의 좌표를 모두 계산하면, 각 이웃셀을 포함하는 클러스터에 접근하여 사용자 수를 계산한다. 이때, 계산된 누적 사용자 수(M′)가 사용자가 원하는 K보다 클 경우, 사용자를 포함하여 탐색된 셀을 cloaking 영역으로 설정하고 알고리즘을 종료한다. 그러나 사용자가 원하는 K보다 작다면, 이웃셀 중 하나를 기준셀로 설정하고 cloaking 영역 설정 알고리즘을 반복한다.

Cloaking 영역 설정 알고리즘

```
//입력: 질의 요청자(q)의 좌표정보, k(cloaking 영역 내에 포함시킬 사용자 수), 힐버트 커브 방향(D), 힐버트 커브 차수(G)
//출력: K를 만족하는 cloaking 영역
01. q 를 기준셀로 설정
02. while( M′(누적 사용자 수) < k )
03. {
04.     for( 이웃셀 좌표 계산 )
05.     {
06.         계산된 좌표를 비트로 표현하고,
            반복횟수 카운트(i)를 0으로 설정
07.         for( i < G )
08.         {
09.             비트를 G-1만큼 이동하여 비트 연산
10.             OHilb(Row, Column)
                //D에 해당하는 힐버트 커브 순서 배열에서 해당 값(V)을 검색
11.             DHilb(Row, Column)
                //D에 해당하는 힐버트 커브 방향 배열에서 V를 검색하여 D에 설정
12.             i++
13.         }
14.         if( i == G )
15.         {
16.             각 G에서 검색한 V를 합산하여 Cj에 저장
17.         }
18.     }
19.     for( 계산된 이웃셀을 모두 탐색 )
20.     {
21.         셀 탐색순서에 따라 계산된 이웃셀(Cj) 선택
            //C_min = Min((H-C1),(H-C2),(H-C3),(H-C4))
22.         클러스터의 finger 테이블을 탐색하여 Cj에 해당하는 클러스터 결정
23.         해당 클러스터에서 Cj와 같은 ID를 갖는 사용자의 수(M)을 검색
24.         누적 사용자수(M′) = M′ + M
25.         if(M′ >= k)
26.         {
27.             탐색된 셀을 cloaking 영역을 결과로 반환하고 알고리즘을 종료
28.         }
29.     }
30.     if(M′ < k)
31.     {
32.         이웃셀 중 하나를 기준셀로 설정
33.     }
34. }
```

그림 11. 힐버트 커브를 이용한 효율적인 cloaking 영역 설정 알고리즘

4.2 Cloaking 영역 설정 알고리즘의 예제

위치기반 서비스는 사람이 많은 도심지에서 주로 이용되므로, 모바일 사용자가 개인 정보를 보호하면서 서비스를 이용하기 위해서는 k값이 최소 5~6이상이 되어야 한다. 따라서 사용하는 예제에서는 사용자를 포함한 k=6인 cloaking 영역을 요청할 경우를 고려한다. 사용자가 cloaking 영역을 요청하면, 먼저 힐버트 커브 그리드 정보 테이블에서 힐버트 커브 방향과 차수를 검색한다(<그림 12>). 예제에서는 힐버트 커브 방향 D가 B타입이고, 차수 G는 3이다.

힐버트 커브 그리드 정보 테이블

사용자식별자 (힐버트 커브 값)	현재 셀 정보	힐버트커브차수	힐버트커브방향
10	C_8	3	B

힐버트 커브 기본 순서 배열

방향 \ 인덱스	0	1	2	3
B	0	3	1	2
T	1	2	0	3
R	2	3	1	0
L	3	2	0	1

힐버트 커브 기본 방향 배열

방향 \ 인덱스	0	1	2	3
B	L^{-1}	R	B	B
T	T	T	L	R^{-1}
R	R	B	R	T^{-1}
L	B^{-1}	L	T	L

그림 12. 사용되는 모바일 기기 저장 구조

사용자의 위치가 C10로 좌표가 (3,3)의 위치에 있다면 (그림 13), 이웃 셀은 표시된 영역의 셀이 되며, 좌표는 각각 (3,2), (2,3), (3,4), (4,3) 이다. 이를 비트로 표현하면 (011,010), (010,011), (011,100), (100,011)이 된다.

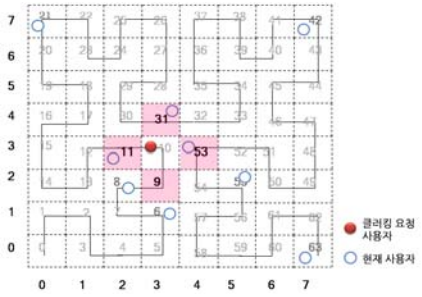


그림 13. 사용자 셀의 이웃 셀을 계산

식(2)에 따라 힐버트 커브 순서 및 방향 배열의 인덱스를 계산하면, 예를 들어 (011,010)의 경우는 다음과 같다.

R0 = ((011 >> 2) & 1) + 2 * ((010 >> 2) & 1) = 0 + 0 = 0,
R1 = ((011 >> 1) & 1) + 2 * ((010 >> 1) & 1) = 3,
R2 = ((011 >> 0) & 1) + 2 * ((010 >> 0) & 1) = 1

인덱스 계산 후 식(3)~(5)에 따라 해당 배열의 인덱스에서 값을 가져오면 각 값은 다음과 같다.

V0 = OHilb(R_0,D_0) = OHilb(0,D) = OHilb(0,B) = 0,
V1 = OHilb(R_1,D_1) = OHilb(3, DHilb(R_0,D_0)) = OHilb(3, DHilb(0,B)) = OHilb(3,L^{-1}) = 2,
V2 = OHilb(R_2,D_2) = OHilb(1, DHilb(R_1,D_1)) = OHilb(1, DHilb(3,L^{-1})) = OHilb(1,L^{-1}) = 1,

이를 서브 그리드의 크기와 곱하여 합산하면 최종적으로 셀의 ID는 다음과 같다.

$$C = V_0 * 2^{2(3-1)} + V_1 * 2^{2(3-2)} + V_2 * 2^{2(3-3)} = 0 * 16 + 2 * 4 + 1 = 9$$

이와 같이 이웃 셀을 계산하면 (3,2), (2,3), (3,4), (4,3)는 각각 9, 11, 31, 53이 된다. 이웃 셀을 방향 별로 계산하면, 이웃 셀에 있는 사용자를 탐색하기 위해 클러스터의 대표노드에 접근하여 각 셀의 클러스터를 검색한다. 이웃 셀 탐색 순서는 사용자 셀과 가까운 ID 순으로 9, 11, 31, 53 이다. 대표 노드의 finger 테이블에는 <그림 14>에서와 같이 N1, N5, N12, N24, N32, N52 가 저장되어 있다.

대표 노드의 finger 테이블

구성원	$C_8(1),C_9(1),C_{10}(1), C_{11}(1)$			
선행자	N5			
후행자	N24			
finger	Entry	후행자 리스트	Entry	선행자 리스트
	N12+1(2^0)	N24	N12-1(2^0)	N12
	N12+2(2^1)	N24	N12-2(2^1)	N12
	N12+4(2^2)	N24	N12-4(2^2)	N12
	N12+8(2^3)	N24	N12-(2^3)	N5
	N12+16(2^4)	N32	N12-16(2^4)	N1
	N12+32(2^5)	N52	N12-32(2^5)	N52

그림 14. 클러스터의 대표노드 저장 구조 예제

셀 9의 클러스터에 해당하는 노드를 검색하기 위해 식 (7)과 (8)을 이용하여 계산하면 다음과 같다.

N1 : 1-9 = -8, ∴ $-8+2^6-1 = -8+63 = 55$
N5 : 5-9 = -4, ∴ $-4+2^6-1 = -5+63 = 58$
N12 : 12-9 = 3
N24 : 24-9 = 5
N32 : 32-9 = 23
N52 : 52-9 = 43

따라서 최소값은 N12이므로 N12를 검색한다. 그러나 그림 13에서처럼 N12에는 C9에 해당하는 사용자가 존재하지 않기 때문에 다음 이웃 셀을 탐색한다. 11은 N12, 31은 N32, 53은 N1이 셀을 포함한 클러스터이므로, 이들에 접근하여 사용자 수를 계산한다. 그러나 k가 일치하지 않기 때문에, 확장할 기준 셀을 가장 먼저 탐색한 이웃 셀인 C9로 설정하고 이웃 셀을 계산한 후, 사용자를 검색한다. 이때 C6, C8에 사용자가 존재하므로, 이를 포함하는 영역은 최종적으로 그림 15의 색칠된 영역이 된다. 설정된 영역은 위치 기반 서비스 서버에 전송된다. 이 예제에서 설정된 영역의 셀의 수는 12개이고, 대표노드가 N1, N12, N32, N52인 클러스터에 접근한다. 그러나 MOBIHIDE 기법(그림 3 참조)에서는 영역 셀의 수는 30이고, 대표노드가 N1, N12, N24, N32, N42, N52인 클러스터에 접근한다. 따라서 본 논문에서 제안하는 알고리즘이 보다 효율적임을 알 수 있다.

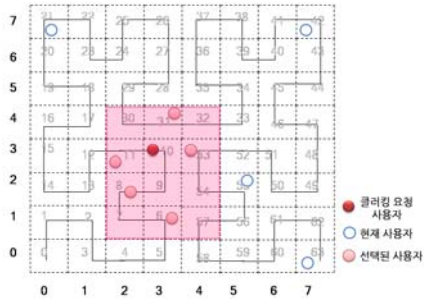
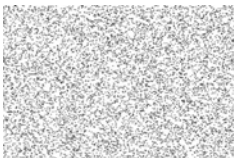


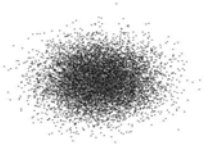
그림 15. 최소 크기의 cloaking 영역 설정 예제

4. 성능평가

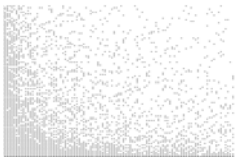
본 논문에서 제안하는 힐버트 커브를 이용한 효율적인 cloaking 영역 설정 기법(이하 성능평가 상에서 DAHC (Direct Accessing of adjacent cells using Hilbert Curves)로 표기)은 Microsoft Visual Studio 6.0으로 구현하였으며, Intel Core2 Duo CPU E4500 2.20GHz와 Ram 2GB, Window XP 상에서 성능을 평가하였다. 성능평가에 사용된 데이터는 GSTD(Generate SpatioTemporal Data) 알고리즘[12]을 이용하였으며, 성능평가를 위해 그림 16과 같이 데이터 분포가 균일, 정규(mean 0.5, variance 0.1), 비대칭(skewedness 1)한 경우를 고려하여 10,000명의 사용자를 생성하고, 질의처리를 수행하기 위하여 100,000개의 POI(Point Of Interest)를 생성하였다. GSTD 알고리즘은 가로 및 세로의 범위가 1인 사각형 내에 객체를 생성하는 방식을 사용하기 때문에, 실험에서 cloaking 영역의 최대 크기는 1로 설정한다. 성능평가 대상은 기존 분산 방식의 cloaking 기법 중 가장 우수한 성능을 보이는 G. Ghinita et al.의 연구인 MOBIHIDE 기법을 구현하여 비교하였다. 성능 평가 항목은 데이터 분포에 따라 k를 20으로 고정한 후, 클러스터 멤버 수를 15에서 200까지 변화시키면서 cloaking 영역 설정 속도를 측정하고, k를 20에서 160으로 변화시키면서 영역 크기 및 영역 설정 속도, cloaking 영역에 따른 질의처리 결과 후보 집합 개수 및 수행 속도를 측정하였다.



(a) 균일(uniform) 분포



(b) 정규(normal) 분포

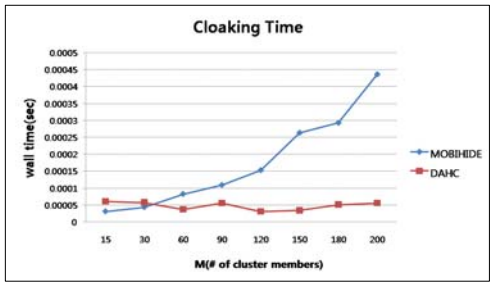


(c) 비대칭(skewed) 분포

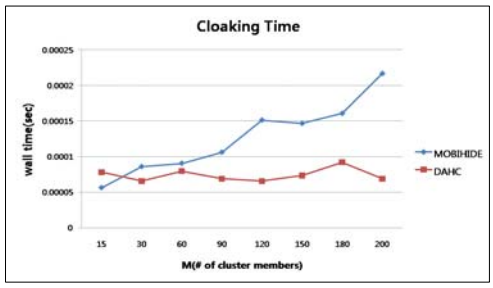
그림 16. 모바일 사용자 분포

4.1 클러스터 내의 구성원 멤버 수(이하 M) 변화에 따른 성능평가

그림 17은 각 데이터 분포를 고려하여 DAHC와 MOBIHIDE의 cloaking 영역 설정 시간을 비교한 것이다. 각 데이터 분포에서 M의 값이 증가함에 따라 MOBIHIDE의 영역 설정시간은 증가하는데 비해, DAHC는 크게 변화하지 않는다. 이는 DAHC가 개선된 finger 테이블을 사용하기 때문에 클러스터 내의 멤버 수가 변화하더라도 다른 클러스터에 최소한의 접근으로 사용자를 탐색하기 때문이다. 한편, M이 15일 경우 DAHC는 MOBIHIDE에 비해 영역 설정 시간이 균일 분포에서 1.95배, 정규 분포에서 1.38배, 비대칭 분포에서 2.2배 더 소요된다. 하지만 M이 120일 경우 DAHC가 균일 분포에서 4.9배, 정규분포에서 2.3배, 비대칭 분포에서 1.4배 빠른 것을 볼 수 있다. 일반적으로 위치기반 서비스는 사람이 많은 도심지에서 주로 이용되기 때문에, 클러스터를 구성하는 사용자의 수가 많은 것이 실제 응용에 보다 적합하다.



(a) 균일(uniform) 분포



(b) 정규(normal) 분포

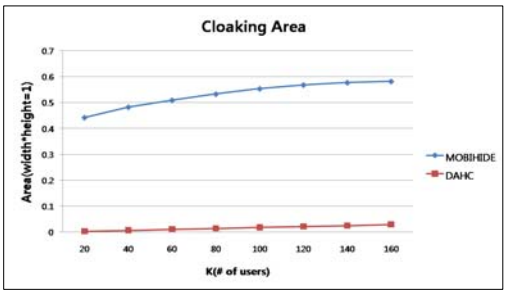
(c) 비대칭(skewed) 분포

그림 17. DAHC와 MOBIHIDE의 cloaking 영역 설정 시간 비교

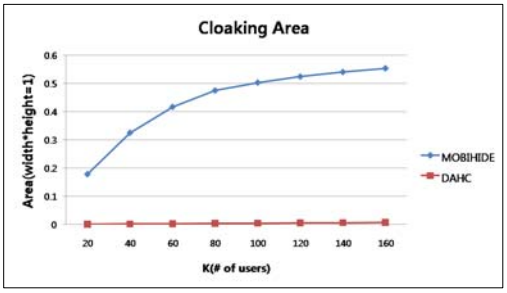
4.2 K-anonymity(이하 k) 변화에 따른 성능평가

k값을 변화하는 실험에서는 실제 응용을 고려하여 M의 값을 120으로 설정한 후 실험을 수행하였다. 그림 18은 데이터 분포를 고려하여 DAHC와 MOBIHIDE의 cloaking 영역 크기를 비교한 것이다. 두 방법 모두 k가

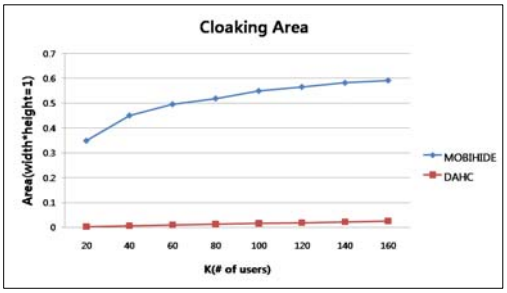
커질수록 cloaking 영역 크기가 증가한다. DAHC는 인접한 셀을 계산하여 cloaking 영역을 설정하므로 k에 만족하는 최소 크기의 cloaking 영역을 설정한다. 반면, MOBIHIDE는 힐버트 커브를 순차적으로 탐색하여 영역을 설정하기 때문에, cloaking 영역 크기가 증가한다. 즉, 균일 분포에서 k가 60일 때 DAHC의 평균 cloaking 영역 크기는 0.0106로, MOBIHIDE의 0.5094보다 약 48배 작고, 정규 분포에서 k가 60일 때 DAHC의 평균 cloaking 영역 크기는 0.0031로, MOBIHIDE의 0.4168보다 약 134배 작다. 또한, 비대칭 분포에서 k가 60일 때 DAHC의 평균 cloaking 영역 크기는 0.0091로, MOBIHIDE의 0.4851보다 약 54배 작다.



(a) 균일(uniform) 분포



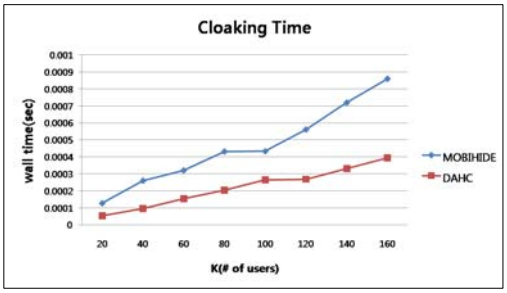
(b) 정규(normal) 분포



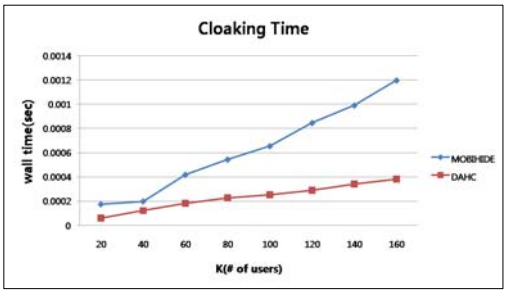
(c) 비대칭(skewed) 분포

그림 18. DAHC와 MOBIHIDE의 cloaking 영역 크기 비교

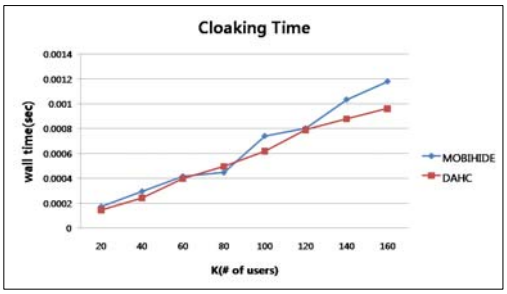
그림 19는 데이터 분포를 고려하여 DAHC와 MOBIHIDE의 cloaking 영역 설정 시간을 비교한 것이다. 균일분포나 정규분포에서는 DAHC가 MOBIHIDE에 비해 평균 2배정도 빠른 영역 설정 속도를 보인다. 이는 DAHC가 개선된 finger 테이블을 사용하기 때문에, cloaking 영역 설정 시 클러스터의 대표노드를 최소로 접근하면서 사용자를 탐색하기 때문이다. 하지만, 비대칭 분포에서는 DAHC와 MOBIHIDE가 유사하게 cloaking 영역을 설정한다. 이는 데이터가 비대칭적으로 분포되기 때문에, 인접셀을 계산하여 cloaking 영역을 설정하더라도, 클러스터의 대표노드에 이웃셀이 존재할 확률이 적어 다수의 클러스터를 탐색할 가능성이 증가하기 때문이다.



(a) 균일(uniform) 분포



(b) 정규(normal) 분포



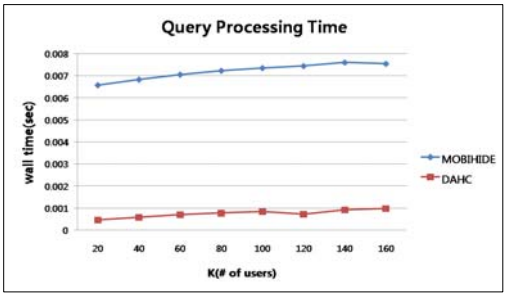
(c) 비대칭(skewed) 분포

그림 19. DAHC와 MOBIHIDE의 cloaking 영역 설정 시간 비교

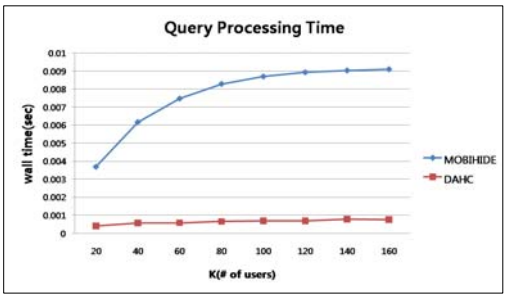
4.3 질의처리 수행시간 성능평가

실제 위치기반 서비스의 이용과정에서 최소 크기의 cloaking 영역 설정이 질의 처리에 미치는 영향을 알아보기 위하여, 설정된 cloaking 영역 크기에 따른 질의처리

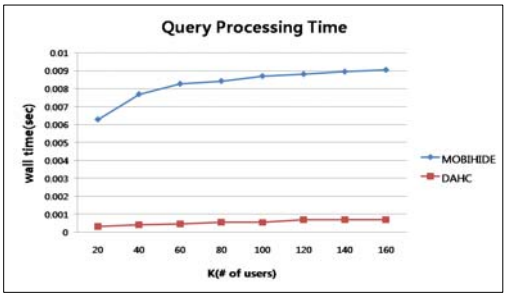
수행 시간 및 결과 후보 집합의 개수를 측정하였다. 질의는 범위 질의(Range Search)를 수행하였다. 그림 20은 데이터 분포를 고려하여 DAHC와 MOBIHIDE의 질의 처리 수행 시간을 비교한 것이다. DAHC는 MOBIHIDE에 비해 각 데이터 분포에서 최소 크기의 cloaking 영역을 설정하기 때문에 서버에서 범위 질의를 수행하는 시간이 적게 소요됨을 알 수 있다. 즉, 균일 분포에서 k가 20일 때 DAHC의 평균 질의처리 수행시간은 0.0004로, MOBIHIDE의 0.0065에 비해 약 14배 빠르고 정규분포에서 k가 20일 때 DAHC의 평균 질의처리 수행시간은 0.0004로, MOBIHIDE의 0.0037에 비해 약 9배 빠르다. 또한, 비대칭 분포에서는 k가 20일 때 DAHC의 평균 질의처리 수행시간은 0.0003로, MOBIHIDE의 0.0063에 비해 약 19배 빠르다.



(a) 균일(uniform) 분포



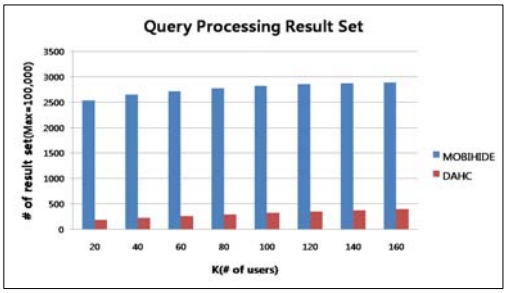
(b) 정규(normal) 분포



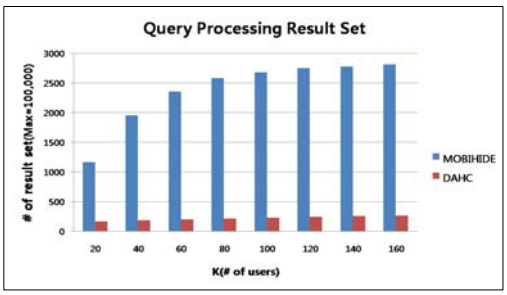
(c) 비대칭(skewed) 분포

그림 20. DAHC와 MOBIHIDE의 질의 처리 수행 시간 비교

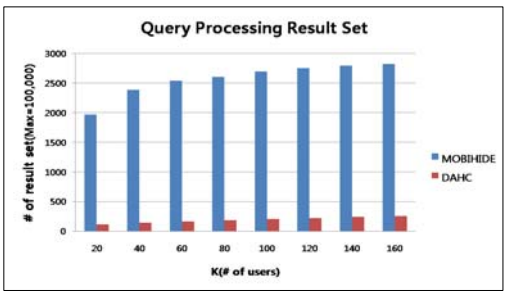
그림 21은 데이터 분포를 고려하여 DAHC와 MOBIHIDE의 질의 처리 결과 집합을 비교한 것이다. DAHC는 각 데이터 분포에서 최소 크기의 cloaking 영역을 설정하기 때문에, 범위 질의시 적은 개수의 후보 집합이 검색된다. 후보 집합의 수가 많을수록 데이터베이스 서버에서 사용자에게 질의 처리 결과를 전송하기 위한 통신비용이 증가하기 때문에, 제안하는 기법이 MOBIHIDE에 비해 효율적임을 알 수 있다.



(a) 균일(uniform) 분포



(b) 정규(normal) 분포



(c) 비대칭(skewed) 분포

그림 21. DAHC와 MOBIHIDE의 질의 처리 결과 집합 비교

5. 결론 및 향후연구

최근 무선 통신과 모바일 위치 측정 기술의 발전으로 위치 기반 서비스(Location-Based Service)의 이용이 확산되었다. 하지만 이러한 서비스는 사용자의 정확한 위치 정보를 가지고 데이터베이스 서버에 서비스를 요청하기

때문에, 심각한 개인 정보 누출의 위협이 될 수 있다. 따라서 모바일 사용자의 안전하고 편리한 위치기반 서비스 사용을 위한 개인 정보 보호 방법이 필요하다.

본 논문에서는 분산 그리드 환경에서 위치 기반 서비스를 이용하는 사용자의 정보 보호를 위해, 힐버트 커브를 이용한 효율적인 cloaking 영역 설정 기법을 제안하였다. 제안한 기법은 사용자가 요구하는 K-anonymity를 만족하는 cloaking 영역을 설정하기 위해, 힐버트 커브의 특성을 분석하고 이를 통해 이웃 셀의 아이디를 계산하여 최소화된 cloaking 영역을 설정한다. 또한, Chord 기반의 분산 해쉬 테이블을 개선하여, 이웃 셀의 클러스터에 해당하는 대표 노드를 빠르게 접근하기 때문에 네트워크의 통신비용을 감소시킬 수 있다. 마지막으로 기존에 제안된 MOBIHIDE와의 성능비교를 통해, 제안하는 기법이 K-anonymity를 만족하는 최소 크기의 cloaking 영역을 설정하고, 빠른 질의 처리 수행 시간을 달성함을 보였다.

향후 연구로는 본 논문에서 제안한 분산 cloaking 기법을 바탕으로 K-최근접 및 궤적질의 처리 알고리즘을 연구하는 것이다.

참 고 문 헌

[1] 이준석, 김서균, “위치기반서비스(LBS)의 기술 동향 및 국내외 산업 동향 분석,” 정보통신연구진흥원 계간, 제 5권 제 2호(통권 16호), 2003.

[2] 김영창, 김영진, 장재우, “대용량 이동객체의 위치정보 관리를 위한 S-GRID를 이용한 분산 그리드 기법,” 한국공간정보시스템학회 논문지, 제 10권 제 4호, 2008, pp. 11-19.

[3] ABC News, Cottonwood man accused of stalking ex-girlfriend with GPS,http://www.abc15.com/content/news/northernarizona/story/Cottonwood-man-accused-of-stalking-ex-girlfriend/4xXXHK-B2FU2B1_Sqzaro9w.cspx, 2008.

[4] Foxs News, Man Accused of Stalking Ex-Girlfriend With GPS, http://www.foxnews.com/story/0,2933,131-487,00.html, 2004.

[5] G. Ghinita, P. Kalnis and S. Skiadopoulos, “MobiHide: A Mobilea Peer-to-Peer System for Anonymous Location-Based Queries,” In Proc. of SSTD, Vol.4605, 2007, pp. 221-238.

[6] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek and H. Balakrishnan, “Chord: A Scalable Peer-to-peer Lookup Service for Internet Application,” In Proc. of IEEE/ACM TON, Vol.11 No.1, 2003, pp. 17-32.

[7] B. Gedik and L. Liu, “Location Privacy in Mobile Systems: A Personalized Anonymization Model,” In Proc. of IEEE ICDCS, 2005, pp. 620-629.

[8] M. F. Mokbel, C. Chow, and W. Aref, “The New Casper:Query Processing for Location Services without Compromising Privacy,” In Proc. of VLDB, 2006, pp. 763 - 774.

[9] 김지희, 이아름, 김용기, 엄정호, 장재우, “위치기반 서비스에서 개인 정보 보호를 위한 K-anonymity 및 L-diversity를 지원하는 Cloaking 기법,” 한국공간정보시스템학회 논문지, 제10권 제4호, 2008, pp. 1-10.

[10] C. Y. Chow, M. F. Mokbel, and X. Liu. A, “Peer-to-Peer Spatial Cloaking Algorithm for Anonymous Location-based Services,” In Proc. of the ACM-GIS, 2006, pp. 171 - 178.

[11] G. Ghinita, P. Kalnis and S. Skiadopoulos, “PRIVE: Anonymous Location-Based Queries in Distributed Mobile Systems,” In Proc. of WWW, 2007, pp. 371-380.

[12] Yannis Theodoridis, Jefferson R. O. Silva, and Mario A. Nascimento, “On the Generation of Spatiotemporal Datasets”, In Proc. of SSD, Vol. 1651, 1999, pp.147-164.

이 아 름

2008년 전북대학교 컴퓨터공학과(공학사)
2008년~현재 전북대학교 컴퓨터공학과 석사과정
관심분야는 공간 데이터베이스, 질의처리, 위치 보안을 위한 cloaking

엄 정 호

2004년 전북대학교 컴퓨터공학과(공학사)
2004년~2006년 전북대학교 컴퓨터공학과 (공학석사)
2006년~현재 전북대학교 컴퓨터 공학과 박사과정
관심분야는 공간 데이터베이스, 공간 색인 구조, GIS

장 재 우

1984년 서울대학교 전자계산기공학과 (공학사)
1986년 한국과학기술원 전산학과 (공학석사)
1991년 한국과학기술원 전산학과 (공학박사)
1996년~1997년 Univ. of Minnesota, Visiting Scholar
2003년~2004년 Penn State Univ., Visiting Scholar.
1991년~현재 전북대학교 컴퓨터공학과 교수
관심분야는 공간 네트워크 데이터베이스, 하부저장구조, 센서 네크워크