멀티무선채널을 갖는 모바일 환경에서 브로드캐스트 데이타를 위한 인덱스 할당 방법

(An Index Allocation Method for the Broadcast Data in Mobile Environments with Multiple Wireless Channels)

이 병 규 † 정 성 원 †† 이 승 중 †††
(Byungkyu Lee) (Sungwon Jung) (Seungjoong Lee)

요 약 브로드 캐스트(Broadcast)는 하나 또는 여러 개의 채널을 이용해서 다수의 이동 단말기들이 빈번하게 필요로 하는 데이타를 효과적으로 전송하기 위한 방법 중에 하나이다. 이동 단말기의 배터리 소모량을 최소화 하기위해서, 브로드캐스트 되는 데이타를 위한 접근시간을 최소화 하는 것이 가장 중요한 문제로 대두되었다. 이런 관점에서 브로드캐스트 되는 데이타에 인덱스를 제공함으로써 데이타 접근시간을 최소화하는데 초점을 맞춘 연구들이 수행되어 왔다. 이 논문에서는 멀티채널(Multi Channels) 환경에서의 효과적인 브로드캐스트(Broadcast)를 위한 인덱스 할당 방법에 대해 연구하였다. 지금까지 제안된 방법들은 브로드캐스트 되는 데이타와 인덱스의 크기가 같아야 한다는 가정을 필요로 하거나 사용되는 인덱스 트리의 깊이와 같은 수의 채널이 주어지지 않을 경우 본래의 성능을 보이지 못하는 단점을 가지고 있다. 앞에서 언급한 두 가지 문제점은 브로드캐스트 데이타에 대해서 평균 접근시간의 증가를 초래하게 된다. 이러한 문제점을 극복하기 위하여 본 논문에서는, 채널을 데이타와 인덱스 채널로 분리한 후, 빈도수가 높은 데이타와 해당 데이타에 대한 트리 구조 인덱스를 보다 많이 브로드캐스트 함으로써 브로드캐스트 데이타에 대한 평균 접근 시간을 최소화하는 방법을 제안한다. 본 논문에서는 제안하는 방법의 실험적 및 이론적인 성능 분석을 이와 비슷한 다른 방법들과 비교함으로써 수행한다. 제시한 방안의 성능 분석은 현존하는 다른 방법과 비교해서 브로드캐스트 데이타에 대해서 현저한 평균 접근 시간 감소를 보여준다.

키워드 : 모바일 컴퓨팅, 모바일 데이타베이스, 브로드캐스트, 멀티채널, 인덱스 할당방법

Abstract Broadcast has been often used to disseminate the frequently requested data efficiently to a large volume of mobile units over a single or multiple channels. Since the mobile units have limited battery power, the minimization of the access time for the broadcast data is an important problem. There have been many research efforts that focus on the improvement of the broadcast techniques by providing indexes on the broadcast data. In this paper, we studied an efficient index allocation method for the broadcast data over multiple physical channels, which cannot be coalesced into a single high bandwidth channel. Previously proposed index allocation techniques either require the equal size of index and data or have a performance degradation problem when the number of given physical channels is not enough. These two problems will result in the increased average access time for the broadcast data. To cope with these problems, we propose an efficient tree-structured index allocation method for the broadcast data with different access frequencies over multiple physical channels. Our method minimizes the average access time for the broadcast data by broadcasting the hot data and their indexes more often than the less hot data and their indexes. We present an in-depth experimental and theoretical analysis of our method by comparing it with other similar techniques. Our performance analysis shows that it significantly decrease the average access time for the broadcast data over existing methods.

Key words : Mobile Computing, Mobile Databases. Broadcast, Multi Channels, Index Allocation

· 본 연구는 서강대학교 산업기술연구소의 지원을 받아 수행되었음
† 정 회 원 : 삼성전자 무선사업부 연구원
bkblue@sogang.ac.kr
†† 종신회원 : 서강대학교 컴퓨터학과 교수
jungsung@ccs.sogang.ac.kr
††† 비 회 원 : 서강대학교 컴퓨터학과
neoleesj@sogang.ac.kr
논문접수 : 2003년 3월 27일
심사완료 : 2003년 11월 5일

1. 서 론

빠르게 발전하고 있는 컴퓨터 하드웨어와 무선 네트워크 기술은 모바일 컴퓨팅 분야의 발전을 가속화 하고 있으며 이에 따라 모바일 기기의 사용이 급증하고 있다. 무선 이동 컴퓨팅 환경에서 이러한 이동 단말기들은 통신비용의 감소뿐만이 아니라 전력소모를 감소시키기 위해서 장시간 동안 끊긴 상태로 있을 것이다. 무선 환경에서, 무선 통신 네트워크상의 많은 량의 데이타를 팜탑과 같은 장비를 갖춘 사용자가 적은 전력으로 빠르게 접근할 수 있도록 구성하는 것은 데이타 관리와 원거리 통신 분야에서 새로운 문제로 대두된다[1,2].

이 문제를 해결하기 위해서 브로드캐스팅이 무선 환경에서 가능한 해결책으로써 제안되었다[3-7]. Acharya는 비정규 분포의 접근 패턴을 갖는 데이타에 대한 성능 향상을 제공하기 위한 방법으로 "Broadcast Disks"를 제안했다. 브로드캐스트 디스크의 기본 개념은 대부분의 클라이언트들이 원하는 데이타 아이템을 다른 것보다 자주 브로드캐스트 하는 것이다. 그러나 Acharya의 방법은 의미적으로 연관성이 있는 데이타에게는 잘 동작하지 않는다. 의미론적인 관계를 갖는 데이타에 기초를 둔 브로드캐스트 기법 또한 제안되었다[4-7].

인덱스를 제공함으로써 브로드캐스트 방법의 향상에 초점을 맞춘 많은 연구가 있었다[8-13,22,23]. 인덱스는 이동 클라이언트가 대부분의 시간동안 대기 상태로 있다가 자신이 원하는 데이타가 브로드캐스트 채널을 통해서 도착하였을 때에 활동 상태로 전환하는 것을 가능하게 한다. 데이타 브로드캐스트의 향상을 위한 대부분의 연구는 두 가지 척도-튜닝 시간과 활성 모드-를 통해서 판별할 수 있는 서비스의 질을 높이는데 목적을 두고 있다. 튜닝 시간(tuning time)은 클라이언트가 원하는 데이타를 얻기 위해 채널을 청취(Listen)하는 시간을 나타낸다. 이 시간 동안 클라이언트는 활성 모드(Active mode)로 동작해야 하기 때문에 이 시간은 단말기의 전력 소모량과 직접적으로 비례한다. 접근 시간(access time)은 클라이언트가 원하는 데이타에 대한 요청을 보낸 후부터 그 데이타를 얻었을 때까지 걸린 시간을 의미한다. Imielinski는 같은 빈도수를 갖는 데이타들이 같은 무선 채널상에서 브로드캐스트 되게 하는 트리 구조의 인덱스 할당방법을 제안했다[9]. 그들은 무선 채널 상에서 인덱스를 통해서 전송되는 데이타에 접근 하는 것이 전력소비의 현저한 감소를 초래한다는 것을 보였다[8,9]. Tan과 Yu는 서로 다른 접근 빈도수를 갖는 데이타를 위한 인덱스 기법을 제안하였다[13]. 그들의 기법은 [9]의 연구를 다양하게 변형시켰다.

Prabhakara 는 멀티 채널을 통해서 이동 사용자에게 데이타를 브로드캐스트 하는 효율적인 방법을 연구하였다[14]. 그들의 방법에서, 그들은 트리구조 인덱스보다도 큰 접근 시간 및 튜닝 시간을 갖는 단순한 구조의 인덱스를 사용하였다. 두개의 서로 다른 접근 방법이 멀티채널 환경에서 데이타를 브로드캐스트하기 위한 트리구조의 인덱스 기법 상에서 연구되었다[10-12]. 첫 번째 접근 방법은 데이타 노드와 인덱스 노드가 각각의 채널에 함께 할당 되도록 멀티채널을 사용하는 것이다. 일반적으로 데이타 노드의 크기는 인덱스 노드의 크기보다 크다(10~50배)는 점을 상기할 필요가 있다. 그러나 첫 번째 방법에서는 인덱스 노드의 크기가 데이타 노드의 크기 때문에 불필요하게 커지게 되며, 이것은 브로드캐스트 데이타에 대한 튜닝시간과 접근 시간을 증가시키는 원인이 된다. 또 다른 접근 방법은, 데이타(인덱스) 채널은 데이타(인덱스)노드만을 전송하기 위해서, 멀티채널을 데이타 전용과 인덱스 전용으로 구분하는 방법이다[6]. 인덱스 채널과 데이타 채널을 구분함으로써, 첫 번째 방법과 같은 심각한 문제는 해결할 수 있었다. 그러나 이 방법은 인덱스 트리의 깊이(depth)보다 물리적인 인덱스 채널수가 적을 때에는 효과적이지 못하다.

본 논문에서, 우리는 멀티채널 상에서 다양한 접근 빈도를 가지는 브로드캐스트 데이타를 위한 새로운 트리구조 인덱스 할당 방법을 제안한다. 본 논문에서 제안하는 방법은 빈도수가 높은 데이타의 인덱스를 빈도수가 낮은 데이타의 인덱스보다 자주 브로드캐스트 함으로써 평균 접근 시간을 최소화시킨다. 이것은 [10,11]에서 제시한 두 가지 방법에서는 가능하지 않다. 또한, 본 논문에서 제시하는 방법은 다음의 두 가지 이유로 인해서 위에서 언급한 두 가지 문제점을 갖지 않는다. 첫째, 본 논문에서는 인덱스 노드와 데이타 노드의 크기가 같아야 하는 가정을 피하기 위해서 [10]에서 제안한 접근 방법을 채택하였다. 둘째로, 본 논문에서 제시하는 방법은 [11]에서 인덱스 트리의 깊이(depth)만큼의 인덱스 채널을 필요로 하지 않는다. 본 논문에서는 새로운 형태의 트리구조 인덱스가 아니라, 접근 시간을 줄이기 위한 트리구조 인덱스를 이용한 새로운 할당 방법을 제공한다.

논문의 나머지 부분은 다음과 같이 구성된다. 먼저 2장에서는 데이타 채널에서의 브로드캐스트 스케줄 방법을 설명한다. 3장에서는 인덱스 채널을 통해서 비정규 분포의 접근 빈도를 가지는 브로드캐스트 데이타를 위한 트리 구조를 이용한 새로운 인덱스 할당 방법을 제안한고 인덱스 채널의 인덱스 노드로부터 데이타 채널의 데이타 노드에 어떻게 접근할 수 있는지를 논의한다. 4장에서는 본 논문에서 제안한 방법의 성능을 분석한다. 5장에서는 최적의 인덱스 채널 개수 및 데이타 채널 개수를 결정하는 방안에 대하여 논의한다. 마지막으로 6장

에서는 결론을 논의 하였다.

2. 데이타 채널에서의 데이타 브로드캐스트 기법

본 논문에서 제안하는 방법은 데이타 채널과 인덱스 채널을 분리한 상태에서의 인덱스 할당 방법이다. 모든 채널들은 대칭적(symmetric)이고, 동일한 대역폭을 갖는 물리채널임을 주의해야 한다. 인덱스 채널과 데이타 채널은 각각 인덱스 노드와 데이타 노드를 전달하여 브로드캐스트 한다. 이 장에서 우리는 먼저 데이타 채널을 통해서 데이타 노드를 브로드캐스트 하는 알고리즘에 대해서 논의한다. 그림 1에서 보이는 것과 같이, 각각의 데이타는 모두 온도를 가지고 있다고 가정한다. 온도가 높은(낮은) 데이타라는 것은 접근 빈도가 높은(낮은) 데이타를 의미한다. 보편성을 잃지 않으면서, 우리는 데이타 채널 i가 데이타 채널 i+1이 가지고 있는 데이타보다 온도가 높은 데이타를 가지고 있다고 가정한다.

그림 1의 알고리즘은 두 단계로 구성된다. 첫 번째 단계는 브로드캐스트 될 데이타 노드들을 각각의 온도에 따라서 내림차순으로 정렬한다. 두 번째 단계에서, 우리는 모든 데이타 노드의 온도 합을 데이타 채널 수로 나눔으로써 평균 채널 온도를 계산한다. 이 단계는 데이타 채널 1에 몇 개의 데이타 노드가 할당되는지를 결정한다. 다시 말해서, 처음 h개의 가장 온도가 높은 데이타 노드의 온도 합이 평균 데이타 채널 온도를 초과하지 않을 때까지 온도가 가장 높은 h개의 데이타를 데이타 채널 1에 할당한다. 이 과정은 채널 2부터 k까지 계속된다. 그러므로 데이타 채널 i에서의 브로드캐스트 주기는 데이타 채널 i+1에서의 주기보다 짧아진다. 이러한 특징은 온도가 높은 데이타 노드가 온도가 낮은 데이타 노드보다 빈번하게 브로드캐스트 되는데, 이 방법으로 인해서 높은 접근 빈도를 가지는 브로드캐스트 데이타의 접근 시간은 감소하게 된다. 이 방법에서, 같은 채널에 할당된 데이타는 같은 브로드캐스트 빈도수를 갖는다고 가정한다.

4개의 데이타 채널을 가지는 알고리즘의 예를 그림 2에서 설명한다. 그림 2는 데이타 온도에 따라서 내림차순으로 정렬된 14개의 데이타를 가지고 있다. 이 데이타 노드는 E, N, A, D, K, L, M, B, C, I, J, F, G, H이고, E가 가장 온도가 높으며, H가 가장 온도가 낮다. 단계2에서, 모든 14개의 데이타의 온도 합을 4로 나눔으로써 평균 데이타 채널 온도를 계산한다. 평균 데이타 채널 온도가 70이라고 가정하자. E와 N의 온도 합이 70을 넘지 않으므로, E와 N은 데이타 채널 1에 위치하게 된다. [A, D, K], [L, M, B, C] 그리고 [I, J, F, G, H]는 각각 데이타 채널 2, 3, 4에 같은 방법으로 위치한다. 이와 같이 데이타를 각 채널에 할당했을 경우, 채널 1을 통한 [E, N]의 브로드캐스트 주기가 데이타

begin
/* **INPUT**:
D = {$d1$, $d2$, ... ,dn} 브로드캐스트 될 데이터 노드들의 집합;
Temperature(di) = 1부터 n까지 데이터 d_i의 온도;
DC ={$DC_1, DC_2, ..., DC_k$} 주어진 데이터 채널들의 집합;
k = 데이터 채널의 갯수
OUTPUT:
DC_i에 있는 데이터 채널을 통해서 D에 속하는 데이터 노드들을 위한 데이터 브로드캐스트 프로그램 */
Step 1:
D에 속하는 데이터 d_i 를 *Temperature*(d_i)에 따라서 내림차순으로 정렬;
내림차순으로 정렬된 상태에서 d_i'가 i번째 데이터라고 가정;
Step 2:
$$ave\_channel\_temperature = \sum_{i=1}^{n} \frac{Temperature(d_i')}{k}$$
j = 0;
for (i = 1; i ≤k; i + +) {
  do {
    j = j + 1;
    데이터 d_i'를 데이터 채널 DC_i에 할당한다.
  } while (데이터 채널 DC_i에 할당된 모든 데이터의 온도 합 < *ave_channel_temperature*);
}
end

그림 1 데이타 브로드캐스트 생성 알고리즘

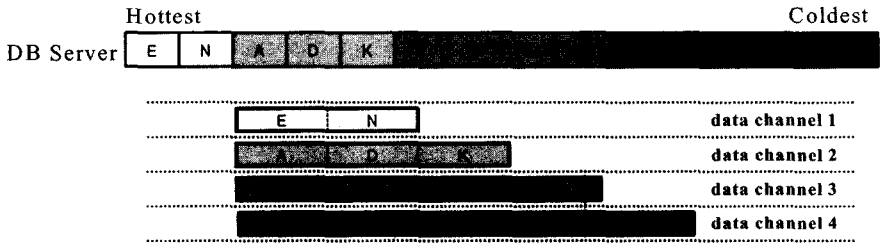


그림 2 데이타 브로드캐스트 스케줄링

채널 2, 3, 4의 브로드캐스트 주기보다 짧아지는 특징을 가지게 된다.

다음 장에서, 우리는 그림 1에서 생성한 데이타 브로드캐스트 스케줄을 반영하는 인덱스 할당 방법에 대해서 논의한다. 데이타 브로드캐스트 스케줄을 반영하기 위해서, 우리의 인덱스 할당 방법은 데이타 채널 i를 통한 데이타의 브로드캐스트 빈도수와 데이타 채널 i의 데이타를 위한 인덱스들의 브로드캐스트 빈도수와 같아지는 것을 요구한다. 이 요구사항은 온도가 낮은 데이타의 인덱스보다 온도가 높은 데이타의 인덱스가 보다 자주 브로드캐스트 되어야 한다는 것을 의미한다. 앞에서 말한 데이타와 인덱스 브로드캐스트 스케줄을 통해서, 멀티채널에서의 비정규 분포의 접근 패턴을 가지는 데이타에 대한 평균 접근시간이 최소화된다.

3. 인덱스 채널에서의 인덱스 할당 방법

이 장에서, 우리는 멀티채널에서 편향된 분포의 접근 빈도를 갖는 브로드캐스트 데이타를 위한 인덱스 할당 기법을 제안한다. 우리의 방법은 온도가 높은 데이타와 인덱스를 온도가 낮은 데이타와 인덱스보다 자주 브로드캐스트 함으로써 접근 시간을 최소화한다. 우리의 방법을 논의하기 전에, 기존에 연구되어온 멀티채널에서 트리기반 인덱스 할당방법의 문제점을 살펴보도록 한다.

3.1 기존의 트리기반 인덱스 할당 방법

멀티채널 환경에서 두개의 다른 접근 방법이 편향된 분포의 접근 빈도를 갖는 브로드캐스트 데이타를 위한 트리기반 인덱스 방법이 연구되어 왔다[10,11]. 이러한 방법들은 데이타에 대한 인덱스 방법으로 Alphabetic 호프만 트리(Huffman tree)를 이용한다. Alphabetic 호프만 트리는 Hu-tucker알고리즘에 의해서 생성된다[10,15]. 이 알고리즘은 먼저 데이타를 접근 빈도에 따라 정렬한 후에 접근 빈도가 낮은 데이타부터 결합하여 트리를 생성하는 상향식 트리 구성의 방법을 사용한다. 인덱스 노드들은 가장 온도가 낮은 두 개의 노드들을 합치는 과정을 반복함으로써 생성된다. 앞의 과정에 의해서 생성된 Alphabetic 호프만 트리상의 각 데이타 노드의 깊이(depth)는 다른 데이타 노드와의 상대적인 온도를 나타낸다. 온도가 높은 데이타는 트리의 상위 부분에 위치하고, 접근 빈도가 낮은 데이타는 트리의 하위부분에 위치하게 된다. 이러한 속성으로 인해서 트리의 하위부분에 위치한 데이타가 트리의 상위 부분에 위치한 데이타 보다 작은 튜닝 시간과 접근 시간을 갖게 된다.

그림 2의 데이타 노드 집합 {A, B, C, D, E, F, G, H, I, J, K, L, M, N}을 이용해서 Alphabetic 호프만 트리를 구성한 예가 그림 3이다. Hu-tucker 알고리즘에 의해서 생성된 인덱스 노드의 집합은 {1, 2, 3, ..., 11, 12, 13}이다. 이 예에서 데이타 노드 E와 N은 가장 높은 온도를, 그리고 F와 G는 가장 낮은 온도를 가지고 있으므로, 각각 트리의 최상위와 최하위에 위치하고 있다. 그림 3에서 그린 Alphabetic 호프만 트리는 이 논문에서 계속 예제로 사용될 것이다.

[10]에서, Shivakumar와 Venkatasubramanian은 데이타(인덱스) 채널이 데이타(인덱스)노드만을 전송하도

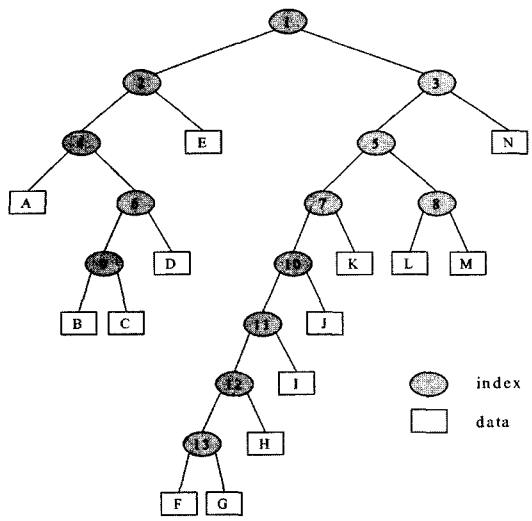


그림 3 Alphabetic 호프만 트리의 예

록 하기 위해서, 멀티채널을 데이타 채널과 인덱스 채널로 구분하였다. 그들은 Alphabetic 호프만 트리의 모든 인덱스 노드들을 각 계층(level) 별로 별도의 인덱스 채널을 사용하여 브로드캐스트 하였다. 이러한 방법은 인덱스 채널의 수가 Alphabetic 호프만 트리의 깊이(depth)보다 작을 경우에 성능이 저하되는 문제를 가지고 있다. 이러한 경우, 그들은 트리의 깊이를 줄이기 위해서 노드의 팬아웃(fanout)을 증가시키거나, 하나의 물리적인 인덱스 채널에 time multiplex를 이용한 가상 인덱스 채널을 사용하였다. 하지만 이러한 방법은 추가적인 연산을 필요로 할 뿐 아니라 브로드캐스트 데이타에 대한 접근 시간의 증가를 가져오게 된다. 그림 4(a)는 그림 3에서 보여준 Alphabetic 호프만 트리의 예를 [6]의 인덱스 할당 기법에 적용시킨 모습이다. 그림 3의 깊이는 8이므로, 그림 4(a)에서는 각 채널이 트리의 각 계층(level)의 인덱스 노드를 의미하는 8개의 인덱스 채널을 갖는다. 이 예는 Alphabetic 호프만 트리의 인덱스 노드들이 편향(skewed)되었을 때 부가적인 문제가 발생함을 보여준다. 온도가 낮은 데이타 {F, G, H, I}를 위한 인덱스 노드 11, 12, 13이 {D, K, L. M}과 같은 온도가 높은 데이타의 인덱스 노드들(예. 6, 7, 8)보다 빈번하게 브로드캐스트 되고 있음을 이 예에서 알 수 있다.

[11]에서, Lo와 Chen은 각 채널에서 데이타 노드와 인덱스 노드가 하나의 채널에 함께 배치되는 멀티채널을 이용하였다. 그들은 브로드캐스트 데이타를 위해서 Alphabetic 호프만 트리를 생성한 뒤, Alphabetic 호프만 트리를 이용해서 주어진 멀티채널에 맞는 위상 트리를 생성하였다. 마지막으로 위상 트리의 인덱스와 데이타 노드를 주어진 멀티채널에 할당하였다. 그림 4(b)는 그림 3의 Alphabetic 호프만 트리로부터 위상 트리를 이용하여 6개의 채널에 데이타와 인덱스를 함께 배치한 모습을 보여준다. 그들의 접근 방법은 인덱스와 데이타가 같은 크기를 가져야 했으므로 문제점을 가지게 된다. 일반적으로 데이타 노드들의 크기는 인덱스 노드들의 크기보다 (10~50배정도) 크다는 점을 주의하여야 할 필요가 있다[16,17]. 인덱스 채널의 크기는 불필요하게 커지고 이는 채널 대역폭의 낭비를 가지고 오며, 브로드캐스트 데이타에 대한 튜닝 시간과 접근 시간의 현저한 증가를 가지고 오게 된다.

이 논문에서, 우리는 멀티채널에서 비정규 분포의 접근 빈도를 갖는 브로드캐스트 데이타를 위한 트리 기반의 새로운 인덱스 할당 기법을 제안한다. 온도가 높은 데이타와 인덱스가 온도가 낮은 데이타와 인덱스 보다 자주 브로드캐스트 함으로써 평균 접근 시간을 최소화한다. 본 논문에서 제안하는 방법은 [10,11]에서 발생한 2가지 문제점을 극복한다. 그림 4(c)는 2개의 인덱스 채널과 4개의 데이타 채널이 주어졌을 때, 이 논문에서의 인덱스 할당기법의 예를 보여준다. 이 논문에서 제안하는 방법을 자세하게 논의하기 전에, 본 논문 전체에 걸쳐 사용될 가정과 용어를 소개한다.

3.2 가정과 용어

이제 본 논문에서 제안하는 방법을 설명하기 위해 필요한 가정들과 용어들을 살펴보도록 하겠다.

- 모든 채널은 모두 대칭적이고, 물리적으로 동일한 대역폭을 갖는다고 가정한다.
- 하나의 채널에서 다른 채널로 변환할 때 걸리는 시간은 무시해도 좋다고 가정한다.
- 데이타 노드의 크기와 인덱스 노드의 크기는 동일하지 않다. 그러나 모든 데이타 노드들의 크기는 동일하며, 모든 인덱스 노드들의 크기는 동일하다고 가정한다.
- 이동 클라이언트들은 자신이 접근해야 하는 채널을 알고 있다고 가정한다. 이동 클라이언트는 한번에 하나의 채널에만 접근한다.
- 인덱스 노드는 채널 번호에 대한 정보와 다음 인덱스 또는 데이타 노드에 대한 시간 오프셋(time offset)을

index channels

1	1	1	1
2	3	2	3
4	5	4	5
6	7	8	6
9	10	9	10
11	11	11	11
12	12	12	12
13	13	13	13

(a) Alphabetic Huffman tree based Method

(index + data) channels

1	2	4	A	9	B
-	3	E	6	D	C
-	-	5	7	10	11
-	-	N	8	K	J
-	-	-	-	L	-
-	-	-	-	M	-

(b) Topological tree based Method

index channels

1	2	...	13	1	2	...	6	1	2	1	2	...	13	...
-	3	...	-	-	3	...	7	-	3	-	3	...	-	...

data channels

E	N	...	E	N	E	...	E	N	E	N	E	...	N	...
A	D	...	A	D	K	...	A	S	K	A	D	...	A	...
L	M	...	L	M	B	...	B	C	L	M	B	...	M	...
J	I	...	H	F	G	...	F	G	J	I	H	...	F	...

(c) Our Index Allocation Method

그림 4 기존의 인덱스 할당방법 및 본 논문에서 제안하는 방법의 예

가지고 있다고 가정한다.

- 접근 시간은 초기 탐색(initial probe)으로부터 이동 클라이언트가 요청한 데이타를 모두 받는 시점까지 경과한 시간이다.
- 인덱스 브로드캐스트 주기 : 모든 인덱스를 브로드캐스트 하는데 걸리는 시간을 의미한다.

그림 5는 접근시간과 인덱스 브로드캐스트의 주기의 예를 보여준다. 그림 5에서 클라이언트는 데이타 7을 얻기 위해서 인덱스 채널에 접근한다. 데이타 7의 인덱스를 얻기 위해서 인덱스 채널 상에서 A만큼의 시간을 소요한다. 그 후, 클라이언트는 데이타 7이 브로드캐스트 될 때 까지 대기 모드(doze mode)에 들어간다. 데이타 7을 모두 받는데 걸리는 시간을 C라고 가정하자. 그러면, 데이타 7에 대한 접근 시간은 A + B + C이다.

3.3 인덱스 할당 방법

이 장에서 우리는 멀티채널을 통한 인덱스 할당 기법을 논의한다. 먼저 본 논문의 뒷부분에서 쓰이게 될 정의를 소개한다.

정의 3.1 DC_i를 i번째 데이타 채널이라 가정하자. 그러면, IS_1은 Alphabetic 호프만 트리의 루트에서 DC_1에 속한 각 데이타까지의 경로(path)에 포함된 인덱스들의 집합이라 정의한다. i번째 인덱스 집합(Index Set) IS_i는 ($2 \le i \le$the number of data channels) 인덱스 트리의 루트에서 DC_i에 속한 각 데이타까지의 경로에 포함된 인덱스 중에서 $IS_j(1 \le j < i)$에 포함된 인덱스를 제외한 인덱스들의 집합이다.

정의 3.1에 의해서 IS_{i+1}을 생성하기 전에 IS_i를 생성해야 한다. DC_i에 속한 데이타의 온도가 DC_{i+1}에 속한 데이타의 온도보다 크므로, IS_i에 속한 인덱스들은 IS_{i+1}에 속한 인덱스보다 온도가 높은 데이타 노드를 가리킨다. 게다가, 접근 시간을 최소화하기 위해서, DC_i에 속한 데이타와 같은 빈도로 IS_i에 속한 인덱스를 브로드캐스트 해야 한다. 이럴 경우, 한 번의 인덱스 브로드캐스트 주기 내에서 반복되어야 하는 IS_i의 반복 빈도를 계산해야만 한다. 다음 정의에서 반복빈도(Replication Frequency) RF_i를 정의한다.

정의 3.2 BD를 브로드캐스트 될 데이타의 개수, DX_i를 IS_i에 속하는 인덱스 노드가 가리키는 데이타의 개수라고 하자. $RF = \left\lfloor \frac{BD}{DX_i} \right\rfloor$는 IS_i가 브로드캐스트 되어야 할 반복 빈도를 나타낸다.

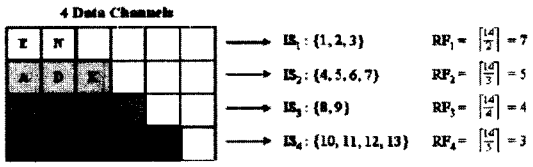


그림 6 인덱스 집합 및 반복빈도

그림 6은 $BD=14$, $DX_1=2$, $DX_2=3$, $DX_3=4$, $DX_4=5$인 그림 3의 Alphabetic 호프만 트리가 갖는 IS_i와 RF_i의 예이다. 위에서 언급한 것처럼, 우리는 IS_1, IS_2, IS_3, IS_4를 순차적으로 생성한다. IS_1 = {1, 2, 3}일 때, RF_1 = 7이며, 이는 IS_1은 한 번의 인덱스 브로드캐스트 주기 안에서 7번 브로드 캐스트 되어야 한다는 것을 의미한다.

다음은 본 논문에서 제안하는 인덱스 할당 방법에서 기본 단위가 되는 인덱스 블록(Index Block)에 대한 정의이다.

정의 3.3 IS_i를 i 번째 데이타 채널 DC_i에 대한 index set이라 하고, 인덱스 채널의 수를 k라 하자.

IS_i의 부분집합의 수열 $S_{i1}, S_{i2}, \ldots, S_{i\in}$은 IS_i에 대

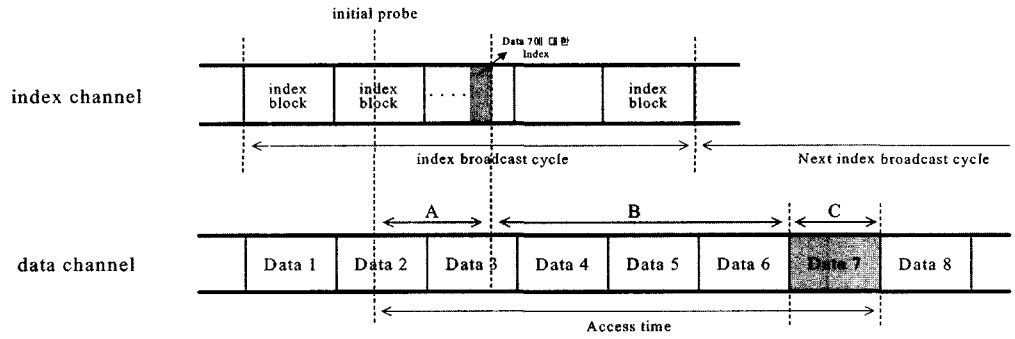


그림 5 인덱스 브로드캐스트 사이클

한 인덱스 블록(index block) IB_i라고 정의한다. 또한, 인덱스 블록 IB_i의 p($1 \le p \le n$)번째 요소를 S_{ip}라고 한다. 인덱스 블록 IB_i는 다음 4가지 사항을 만족시켜야 한다.

1. $IS_i = \bigcup_{j=1}^{n} S_{ij}$
2. $1 \le p, q \le n$ 그리고, $p \ne q$에 대해서 $S_{ip} \cap S_{iq} = 0$
3. S_{ip}에 속하는 모든 인덱스는 서로 조상과 자손(ancestor-descendent) 관계를 가져서는 안된다.
4. $1 \le p \le n$에 대해서 $|S_{ip}|$는 S_{ip}의 크기를 나타내며, $|S_{ip}|=k$이어야 한다. 만약 $|S_{ip}|<k$라면, 크기가 k가 될 때까지, null index를 S_{ip}에 넣는다.

그림 7은 인덱스 채널의 개수가 2일 때, 그림 3의 Alphabetic 호프만 트리로부터 얻은 인덱스 집합 {1, 2, 3}, {4, 5, 6, 7}, {8, 9}, {10, 11, 12, 13}에 대한 인덱스 블록의 예를 보여준다. 두 개의 인덱스 채널을 가지고 있으므로 인덱스 블록의 각 요소의 개수는 2이어야 하고 각 요소에 속하는 모든 원소는 조상 및 자손(ancestor-descendent) 관계를 가져서는 안 된다. 앞에서 언급한 사항으로 인해서 $S_{11}=\{1\}$, $S_{41}=\{10\}$, $S_{42}=\{11\}$, $S_{43}=\{12\}$, $S_{44}=\{13\}$은 자신의 원소로 null 인덱스가 포함되게 된다.

이제 본 논문이 제시하는 인덱스 할당 방법에 대해서 논의하자. $S_{ip} = I_1, I_2, \ldots, I_k$는 정의 3.3에 의해서 인덱스 집합 IS_i에 대한 인덱스 블록의 일부라고 가정한다. 여기서 k는 인덱스 채널의 개수를 나타내고, $I_1, I_2, \ldots I_k$는 각각의 인덱스를 나타낸다. k인덱스들은 조상 및 자손관계를 가지지 않으므로, 데이타 접근에 대한 어떤 순서에 의해서 접근될 필요가 없다. 이러한 속성은 k인덱스가 k인덱스 채널에서 동시에 브로드캐스트 될 수 있도록 한다. 이러한 속성은, 우리가 인덱스 I_j를 인덱스 채널 j ($1 \le j \le k$)에 할당하게 한다. 그림 8은 그림 7의 인덱스 블록의 인덱스들이 두개의 인덱스 채널에 어떻게 할당되는지를 보여주는 예제이다. 지금까지, 우리는 k개의 멀티채널에 인덱스 할당기법에 대해서 논의하였다. 이것은 공식적으로 그림 9에 설명하였다.

위의 알고리즘이 어떻게 동작하는지를 보이기 위해서, 4개의 데이타 채널과 2개의 인덱스 채널을 가지고 있는 예제를 그림 10에서 설명한다. 그림 2의 데이타 브로드캐스트 스케줄과 그림 3의 Alphabetic 호프만 트리을 가정한다. 위의 가정에 근거하여, 이 예제는 인덱스가 2개의 인덱스 채널에 할당되는 모습을 보여준다. 그림 6에서 설명한 반복 주기로써, 인덱스 집합 $IS_1=\{1,2,3\}$, $IS_2=\{4,5,6,7\}$, $IS_3=\{8,9\}$, $IS_4=\{10,11,12,13\}$는 그

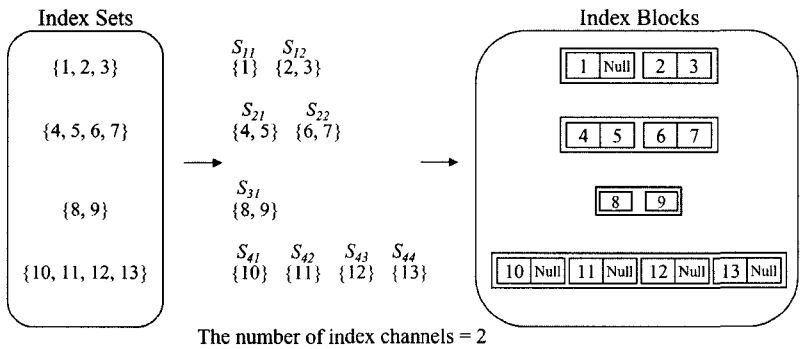


그림 7 인덱스 블록의 예

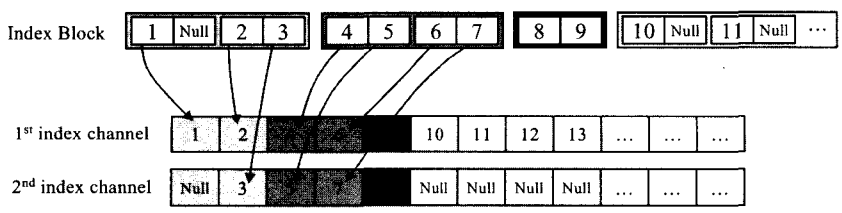


그림 8 인덱스 블록의 인덱스를 2개의 인덱스 채널에 할당

```
begin
/* INPUT :
AHT = 브로드캐스트 데이터에 의해서 생성된 Alphabetic Huffman tree
ND = 데이터 채널의 개수
IC={IC1, IC2, ...., ICk} k 인덱스 채널의 집합
DBP = 그림 2.1의 알고리즘에 의해서 계산된 데이터 브로드캐스트 프로그램
OUTPUT:
DBP를 위한 IC안의 k 인덱스 채널을 통한 인덱스 할당 프로그램 */

Step 1: 인덱스 집합 생성
    for (i=1;i≤ ND;i++) 정의 3.1에 의해서 AHT와 DBP로부터 ISi를 생성
Step 2: 인덱스 집합을 위한 반복 빈도 계산
    for (i=1;i≤ ND;i++) 정의 3.2에 의해서 ISi 로부터 RFi 계산
Step 3: 인덱스 집합을 위한 인덱스 블록 생성
    for (i=1;i≤ ND;i++) 정의 3.3에 의해서 ISi로부터 인덱스 블록 IBi생성
Step 4: IC에서 k인덱스 채널에 인덱스를 할당
    count = 0;
    for (i=1;i≤ ND;i++) count = count + RFi;
    while (count > 0) {
        for (i=1;i≤ ND;i++) {
            if (RFi ≥ 0) {
                IBi가 Si1, Si2, ...Sin으로 구성된다고 하면;
                for (j = 1; j ≤ n; j ++) {
                    Sij = {I1, I2, ..., Ik}라고 가정;
                    for p=1부터 k까지, 인덱스 Ip를 인덱스 채널 ICp에 할당;
                }
                RFi = RFi - 1;
                count = count - 1;
    }   }   }
end
```

그림 9 k인덱스 채널에 대한 인덱스 할당 알고리즘

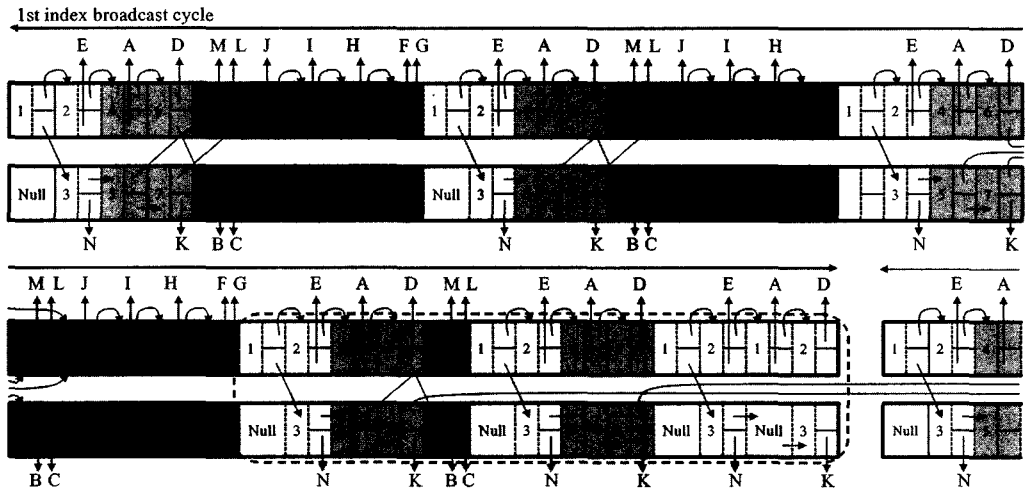


그림 10 그림 9의 알고리즘에 의해서 생성된 인덱스 할당의 예

림 10에서 한 번의 인덱스 브로드캐스트주기 동안에 각각 7, 5, 4, 3번씩 반복된다. 이 예에서, 수평 화살표는 다음 인덱스 노드에 대한 포인터를, 수직 화살표는 데이타 노드로의 포인터를 형상화하였다. 예를 들어, 클라이언트가 데이타 "H"에 접근하고자 한다면, 클라이언트는 1→3→4→7→10→11→12→H 와 같이 7개의 인덱스 노드

와 1개의 데이타 노드에 접근해서 원하는 데이타를 얻을 것이다.

3.4 인덱스 노드를 통한 데이타 노드 접근

이 장에서 우리는 인덱스 채널의 인덱스 노드로부터 데이타 채널의 데이타 노드에 접근하는 방법에 대해서 논의한다. 인덱스 노드는 채널 번호와 자식 인덱스 또는 데이타 노드에 대한 정보를 가지고 있다고 가정한다. 세부적으로, 인덱스 노드 y는 데이타 채널 i를 통해서 전송되는 자식 데이타 노드 x의 위치를 계산하기 위해서 다음의 정보를 알아야만 한다.

- *DSize* : 데이타 노드의 크기
- *Bcast(i)* : 데이타 채널 i를 통해서 브로드캐스트 주기를 구성하고 있는 데이타 노드의 개수
- Size_of_Data_Channel(i) : 데이타 채널 i에 대해서 $BCast\,i \times DSize$
- Relative_Data_Position (x, i) : 데이타 채널 i에서 데이타 브로드캐스트 주기의 시작점으로부터 데이타 노드 x까지의 시간 오프셋.

위의 정보를 기본으로, 인덱스 노드 y에서부터 데이타 채널 i의 데이타 노드 x의 현재 위치를 계산하는 그림 11의 알고리즘을 설명한다. 위의 알고리즘을 설명하기 위해서, 우리는 인덱스 채널과 데이타 채널의 수가 각각 2개와 4개인 그림 12의 예를 사용한다.

인덱스 채널에서 빗금이 있는 상자는 인덱스 노드를 가리키는 인덱스 노드를 나타낸 것이고, 알파벳이 있는 상자는 알파벳이 표시하는 데이타 노드를 가리키는 인덱스 노드를 표시한 것이다.

데이타 노드의 크기가 인덱스 노드의 크기의 10배라 가정한다. 인덱스와 데이타 노드의 사이즈는 시간 단위 t에 의해서 표현되었다. 그러므로 데이타와 인덱스 노드의 위치는 시간 단위에 의해서 구분된다. 다음 예제에서, 2개의 인덱스 채널과 4개의 데이타 채널을 통해서 $t=0$일 때 인덱스 노드와 데이타 노드를 브로드캐스트하기 시작한다. 이동 클라이언트는 데이타 채널 1을 통해서 브로드캐스트 되는 데이타 노드 E에 접근하기 원한다고 가정하자. 데이타 노드 E에 접근하기 위해서, 클라이언트는 첫 번째로, 인덱스 채널의 루트(root) 인덱스 노드에 접근해야 한다. 그 뒤에 클라이언트는 데이타 노드 E를 가리키는 인덱스 트리에 도달할 때 까지 인덱스 트리를 탐색한다. 이 예제에서, 인덱스 노드는 2이고, 이는 그림 12의 인덱스 채널 1에서 E로 표시되고 있다. 인덱스 노드 2가 인덱스 채널 1을 통해서 $t=13$일 때 브로드캐스트 된다고 가정하자. 인덱스 노드 2는 데이타 노드 E가 데이타 채널 1을 통해서 브로드캐스트 되고 있다는 정보를 가지고 있다. 알고리즘을 위한 입력 값은 아래와 같다.

- $x=E,\ y=2, i=1, DSize=10, BCast(1)=2$
- *Size_of_Data_Channel (1)* = *BCast (1)* × *DSize* = 2 × 10 = 20
- *current_index_position* = 13 ($\because$인덱스가 $t=13$에 위치하므로)
- *Relation_Data_Position (E, 1)* = 0

위의 알고리즘으로부터, period_number = 1, position = 20 × 1 + 0 = 20을 알게 되었다. *current_index_position* = 13은 position = 20보다 작은 값이고, while 루프는 position의 값이 20이 되면 실행을 멈춘다. 이것

```
begin
/* INPUT:
  x = 접근하려는 데이터 노드 개수;
  y = 데이터 노드 x를 가리키는 인덱스 노드;
  i = 데이터 노드 x를 브로드캐스트 하게 될 데이터 채널 번호;
  current_index_position = 인덱스 노드 y의 현재 위치;
  Size_of_Data_Channel (i) = BCast (i) × DSize ;
  OUTPUT:
  position = 인덱스 노드 y로부터 데이터 채널 i를 통해서 전송되는 데이터 노드 x의 현재 위치 */

  period_number = ⌊ current_index_position / DSize⌋ ;
  position = Size_of_Data_Channel(i) × period_number + Relative_Data_Position (x, i);
  while (current_index_position > position) {
     period_number ++;
     position = Size_of_Data_Channel(i) × period_number + Relative_Data_Position (x, i);
  }
end
```

그림 11 인덱스 노드로부터 데이타 노드 접근

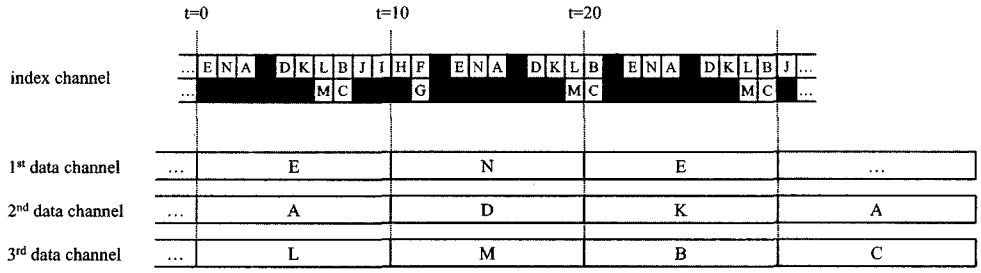


그림 12 멀티채널에서 인덱스 노드를 통한 데이타 노드 접근 예

은 데이타 노드 E가 $t=20$일 때 브로드캐스트 될 것이라는 것을 의미한다. 클라이언트가 데이타 노드 E의 위치를 얻은 뒤에 클라이언트는 20이 될 때 까지 대기 모드에 들어간다. $t=20$에서 클라이언트는 활동모드로 들어간 뒤에 데이타 채널 1로부터 데이타 노드 E를 얻게 된다.

4. 성능 분석

이 장에서는, 위상 트리 기반의 방법[8]과 [6]에서 제시한 Alphabetic 호프만 트리 기반의 방법과 비교를 통해서 본 논문이 제안하는 방법의 성능을 분석한다. 편의를 위해서, 이후에는 이 논문에서 주장하는 방법을 SIRAH(Separated Index Replication based on Alphabetic Huffman tree), 위상트리 기반의 방법을 TOPO(TOPOlogical), 그리고 Alphabetic 호프만 트리 기반의 방법을 ALH(ALphabetic Huffman)라고 명칭한다.

본 논문의 모든 실험은 Pentium-Ⅲ 866EB 프로세서와 512MB 메모리를 갖춘 컴퓨터에서 수행되었다. 시뮬레이터는 CSIM18 시뮬레이션 엔진을 이용해서 구현하였다[18,19]. 이동 클라이언트들의 비정규 분포의 접근 패턴을 모델링하기 위해서 파라미터 θ를 갖는 Zipf 분포를 사용하였다. Zipf 분포는 편향된 분포의 접근 패턴을 모델링 하는 데 일반적으로 사용된다[15,20,21]. θ가 증가할수록 Zipf 분포는 보다 급격히 편향된 접근 패턴을 생성한다. 이 시뮬레이션에서, 편향된 접근 패턴을 모델링하기 위해서 θ의 값을 0.95로 설정하였다. 이 접근 패턴은 브로드캐스팅 환경에서는 일반적이다. 본 논문의 시뮬레이션에서 사용된 여러 가지 파라미터를 표 1에 도표화하였다.

본 논문에서는 이동 클라이언트의 평균 접근 시간을 가장 중요한 성능 지표로 삼았다. 평균 접근 시간은 20000번의 시뮬레이션을 통해서 이동 클라이언트의 평균 접근 시간으로 측정하였다. 접근 시간은 브로드캐스트 단위를 이용하여 측정하였다. 브로드캐스트 단위는 하나의 인덱스 노드를 브로드캐스트 하는 데 걸리는 시간을 나타낸다. 예를 들어, 표 1의 R이 20이라면, 하나의 데이타 노드를 브로드캐스트 하는 데에, 인덱스 노드 20개를 브로드캐스트 하는 시간이 걸린다. 다음의 장에서 여러 가지 조건에서 SIRAH, ALH, 그리고 TOPO의 성능을 분석한다.

표 1 시뮬레이션 파라미터

시뮬레이션 파라미터	범위
데이타 채널의 수	8
인덱스 채널의 수	2
전체 채널의 수	10
브로드캐스트 데이타의 수	2000
$R=\frac{\text{데이터 노드 크기}}{\text{인덱스 노드 크기}}$	1~50
θ	0.95
시뮬레이션 횟수	20000

4.1 데이타 노드 크기와 인덱스 노드 크기의 비율에 따른 효율성

인덱스 노드의 크기는 데이타 노드의 크기와 비교해서 일반적으로 매우 작기 때문에[15,16], 인덱스 노드와 데이타 노드의 크기 차이가 SIRAH, ALH, 그리고 TOPO의 성능에 영향을 미친다. 그러므로 우리는 데이타 노드 크기와 인덱스 노드 크기를 1에서 50까 다양하게 함으로써(i.e., R) 세 가지 인덱스 방법의 성능 효율을 분석한다. 실험 결과는 그림 13이다.

그림 13은 SIRAH가 ALH와 TOPO보다 성능이 좋음을 나타낸다. R이 3보다 작을 경우에만 TOPO가 SIRAH보다 좋은 성능을 보였다. 이것은 멀티 채널 환경에서 데이타 노드 크기가 인덱스 노드 크기와 비슷할 경우에 TOPO가 적합하다는 것을 의미한다. 그러나 앞에서 말한 상황은 멀티채널 브로드캐스팅 환경에서 자주 발생하지는 않는다. 그림 13에서, R이 1에서 20일 때, SIRAH의 평균 접근 시간이 거의 같으며, 20부터

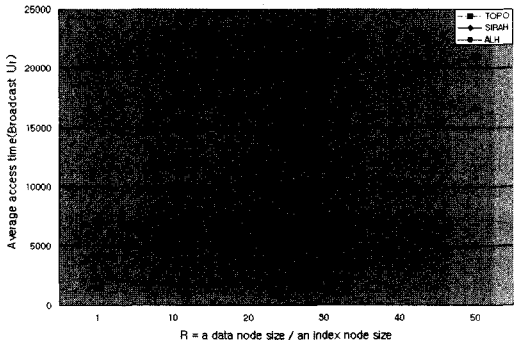


그림 13 R을 변화시켰을 때 TOPO, SIRAH, ALH의 성능

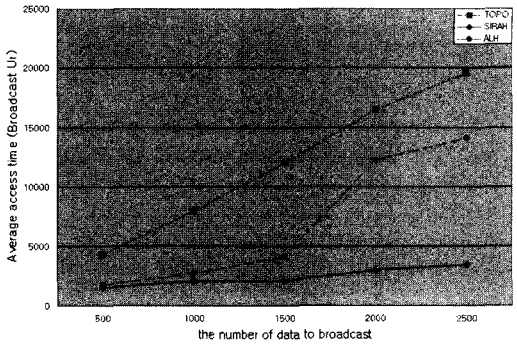


그림 14 인덱스 노드 크기와 데이타 노드 크기의 비율 = 30

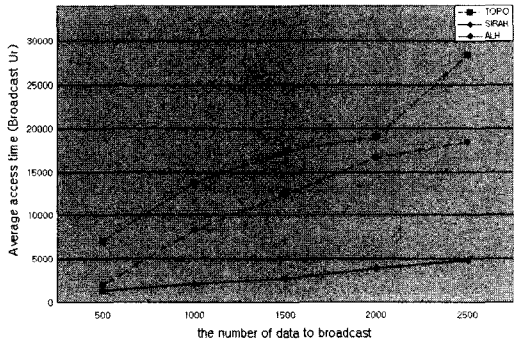


그림 15 인덱스 노드 크기와 데이타 노드 크기의 비율 = 50

50까지도 경사가 완만하게 평균 접근 시간이 증가하는 모습을 보여준다. 이것은 SIRAH의 안정적인 성능을 보여준다. TOPO에서, R이 증가함으로써, 데이타 노드 크기와 인덱스 노드 크기 사이의 차이는 점점 커지게 되고, 그 결과 인덱스 채널에 할당된 대역폭을 낭비하게 된다. 그러나 SIRAH에서는 인덱스 채널과 데이타 채널을 구분하였기 때문에 대역폭 낭비의 문제가 발생하지 않는다.

ALH 방법 역시 인덱스 채널과 데이타 채널을 SIRAH와 같이 구분하였기 때문에 TOPO보다 좋은 성능을 보인다. 위의 결과는 인덱스 노드의 크기가 데이타 노드의 크기보다 커질 때, 인덱스 전용 채널과 데이타 전용 채널을 구분시킴으로써 얻을 수 있는 장점을 보여준다. 그러나 TOPO보다 증가한 성능향상은 SIRAH에서 얻는 성능향상에 비해서 작은 값을 가진다. 이것은 SIRAH 방법과는 다르게, ALH는 인덱스 채널의 수가 인덱스 트리의 깊이(depth)보다 작은 값을 가짐으로써 발생하는 성능 저하 현상과, 인덱스의 브로드캐스트 빈도가 데이타의 접근 빈도를 SIRAH와는 다르게 충분히 반영하고 있지 못하기 때문이다.

4.2 브로드캐스트 데이타의 수에 따른 효율성

이 장에서는, 브로드캐스트 데이타의 개수가 증가하였을 때 SIRAH와 TOPO, 그리고 ALT의 성능에 어떠한 영향을 미치는 지에 대해서 논의해본다. 이 실험에서, R을 30과 50으로 고정시킨 뒤, 브로드캐스트 데이타 수를 500개에서 2500개로 변화를 주었다. 이 실험의 결과는 그림 14와 15이다.

브로드캐스트 데이타의 수를 증가시켰을 때, SIRAH은 TOPO와 ALH보다 훨씬 작은 접근 시간을 갖는다.

이것은 데이타의 크기가 점점 커지면서, 편중된 데이타 접근 패턴이 TOPO와 ALH의 성능을 저하시키기 때문이다. TOPO와 ALH 방법에서는, 높은 온도를 갖는 데이타의 브로드캐스트 주기와 해당 데이타에 대응하는 인덱스 노드가 편중된 데이타 접근 패턴을 충분히 반영하지 못한다.

이러한 현상은, SIRAH와 같이 인덱스 채널과 데이타 채널이 구분되어 있는 ALH에서 보다 명백하게 나타난다. 결과적으로, 그림 14와 그림 15의 TOPO와 ALH방법의 접근 시간은 급격히 증가한다. 반면에, SIRAH방법은 하나의 브로드캐스트 주기 안에서 온도가 낮은 데이타 노드와 인덱스 노드보다 온도가 높은 데이타 노드와 그에 대응하는 인덱스 노드를 보다 빈번히 브로드캐스팅 함으로써 이러한 문제를 극복한다. 그러므로 그림 14와 그림 15에서 보이는 SIRAH의 평균 접근 시간은 매우 완만하게 증가하며, 이를 통해서 매우 안정적인 성능을 보여준다.

4.3 인덱스 채널과 데이타 채널의 수에 따른 효율성

지금까지는, 인덱스 채널 수와 데이타 채널 수가 주어진다고 가정을 하였다. 이 장에서, 데이타 채널과 인덱스 채널이 다양한 값을 가질 때, 시뮬레이션 상에서 TOPO, ALH, 그리고 SIRAH방법의 성능을 분석한다. 이 시뮬레이션에서 사용한 파라미터를 표 2에 도식화하였다.

표 2 데이타 채널과 인덱스 채널 개수의 효율성 분석을 위한 파라미터

시뮬레이션 파라미터	값
전체 채널의 수	2, 3, 4, 5, 10
브로드캐스드 데이타의 수	2000
$R=\frac{\text{데이터노드크기}}{\text{인덱스노드크기}}$	30
θ	0.95
시뮬레이션 총 횟수	20000

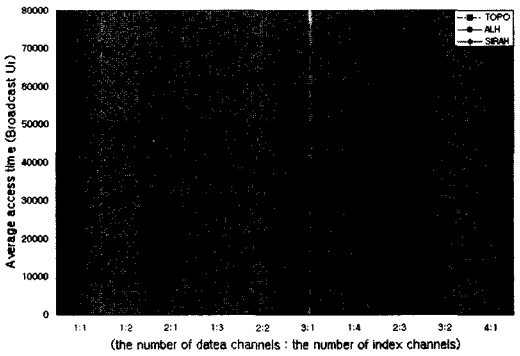


그림 16 채널의 수 = 2,3,4,5

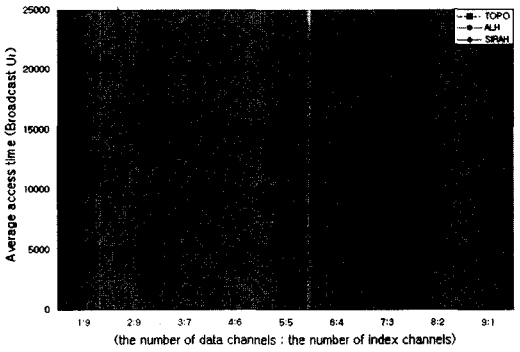


그림 17 채널 수 = 10

그림 16과 그림 17은 채널의 개수가 각각 2, 3, 4, 5 그리고 10일 때의 이동 클라이언트의 평균 접근 시간을 보여준다. 각 채널 수에 대해서, 우리는 가능한 모든 데이타 채널과 인덱스 채널의 순서쌍[a,b]에 대해서 평균 접근 시간을 보여준다. a와 b는 각각 데이타 채널과 인덱스 채널의 수를 나타낸다. 일례로, 채널 수 가 4이면, 본 논문에서는 [1:3], [2:2] 그리고 [3:1]의 경우에 측정을 하였다.

그림 16과 그림 17은 채널 수가 2일 때에만 ALH가 가장 좋은 성능을 나타냄을 보이고 있다. TOPO 방법은 인덱스 채널과 데이타 채널의 구분을 필요로 하지 않으므로, 실험 결과는 데이타와 인덱스 채널의 구성과는 무관한 결과를 갖는다. SIRAH방법은 인덱스 채널과 데이타 채널이 적절하게 구분되었을 때, 가장 좋은 실험 결과를 보인다. 예를 들면, 위의 두 그림을 통해서 전체 채널의 수가 3, 4, 5, 10일 때, 데이타 채널과 인덱스 채널의 최적 채널 수는 각각 [2:1], [3:1], [4:1], [8:2]이다. 위의 예로부터, SIRAH방법의 중요한 속성을 관찰할 수 있다. 최적의 성능을 내기 위해서는 적은 수의 인덱스 채널만을 필요로 한다. 데이타 노드 크기에 비해서 인덱스 노드 크기는 상대적으로 매우 작으므로, 적은 수의 인덱스 채널의 대역폭에 SIRAH 방법으로 구성한 전체 인덱스 트리를 충분히 브로드캐스트 할 수 있다. 결론적으로 나머지 채널의 수는, 데이타 노드를 브로드캐스트 하는데 사용될 수 있으며, 이러한 방법을 통해서 이동 클라이언트의 평균 접속 시간을 줄일 수 있다.

그러면, n개의 채널이 주어졌을 때, 최적의 데이타 채널수와 인덱스 채널수를 결정하는 방법이 문제가 된다. 다음 장에서 이러한 최적의 분할을 결정하는 효율적인 방법을 설명한다.

5. 최적의 데이타 채널 개수와 인덱스채널 개수 결정 기법

이 장에서, 우리는 주어진 브로드캐스트 채널들에서 데이타와 인덱스 채널을 최적의 수로 구분하는 방법에 대해서 논의한다. 인덱스 채널과 데이타 채널의 수는 브로드캐스트 되는 데이타에 대해서 이동 클라이언트의 평균 접근 시간이 최소가 되도록 결정해야 한다. 먼저, 주어진 데이타와 인덱스 채널에서 브로드캐스트 되는 데이타를 위한 평균 접근 시간 $AveAT$의 수식 분석을 보인다. 그 뒤에 평균 접근 시간 $AveAT$를 최소화함으로써 데이타 채널과 인덱스 채널의 최적 구성을 결정하도록 한다. $AveAT$는 평균 인덱스 접근 시간인 IXT와 평균 데이타 접근 시간인 DT로 이루어져 있다. IXT와 DT를 구성하기 전에, 이 장에서 쓰일 파라미터들에 대해서 살펴본다.

- c_d = 데이타 채널의 수
- c_i = 인덱스 채널의 수
- $c = c_d + c_i$, c는 물리적인 채널의 개수
- nj = 데이타 채널 DC_j를 통해서 브로드캐스트 되는 데이타들의 수
- $DSize$ = 데이타 노드의 크기
- $ISize$ = 인덱스 노드의 크기
- $|IS_j|$ = 인덱스 집합 IS_j에 속한 인덱스 노드의 수
- $RF = \sum_{j=1}^{c_i} RF_j$, RF_j는 IS_j의 반복 빈도를 나타낸다.

위에서 열거한 파라미터를 이용해서, 다음의 정리 5.1

에서 IXT와 DT에 대한 비용 모델을 제시다.

정리 5.1 $AveAT$를 c_i개의 인덱스 채널과 c_j개의 데이타 채널을 통해서 브로드캐스트 되는 브로드캐스트 데이타의 평균 접근 시간이라고 하자. 그러면, $IXT+DT$ 이고

$$IXT = \frac{ISize}{2 \cdot c_i \cdot RF_i} \sum_{j=1}^{c_d} RF_j |IS_j| + \frac{ISize}{c_i} \sum_{j=1}^{c_d} \left[\left\{ \sum_{k=1}^{i-1} |IS_k| + \frac{|IS_j|}{2} + \frac{1}{RF_j} \sum_{k=1}^{i-1} |IS_k| (RF_k - RF_j) \right\} \frac{RF_j}{RF} \right]$$

$AveAT = DT = \frac{DSize}{2} \sum_{j=1}^{c_d} \frac{RF_j}{RF} n_j$ 이다.

증명 : case 1 : 먼저 IXT를 증명한다. IXT는 P와 L로 이루어져 있다. P는 초기 탐색(initial probe)로부터 인덱스 트리의 루트에 도달하는 데 걸리는 평균시간을, L은 루트로부터 원하는 leaf인덱스 노드에 도달하는 데 걸리는 평균시간을 나타낸다. P를 계산하기 위해서, 먼저 한번의 인덱스 브로드캐스트 주기 안에서의 인덱스 노드의 전체 개수를 알아야 한다. 그것은 $\sum_{j=1}^{c_d} RF_j |IS_j|$ 로 표현된다. 루트 인덱스 노드는 단일 인덱스 브로드캐스트 주기 상에서 RF_1번 나타나고, 모든 인덱스 노드는 인덱스 채널 c_i에 위치하므로, $P = \frac{ISize}{2 \cdot c_i \cdot RF_1} \sum_{j=1}^{c_d} RF_j |IS_j|$로 나타낸다. 이제,

$L = \frac{ISize}{c_i} \sum_{j=1}^{c_d} \left[\left\{ \sum_{k=1}^{i-1} |IS_k| + \frac{|IS_j|}{2} + \frac{1}{RF_j} \sum_{k=1}^{i-1} |IS_k| (RF_k - RF_j) \right\} \frac{RF_j}{RF} \right]$

임을 보여야 한다. 각각의 인덱스 집합 $IS_1, IS_2, \ldots, IS_{c_d}$ 상에서 인덱스 노드를 얻는 데 걸리는 평균 시간은 다음과 같다.

- IS_1: $\frac{|IS_1|}{2} ISize$
- IS_2: $\left\{ |IS_1| + \frac{|IS_2|}{2} + \frac{1}{RF_2} |IS_1| (RF_1 - RF_2) \right\} ISize$
- IS_3: $\left\{ |IS_1| + |IS_2| + \frac{|IS_3|}{2} + \frac{1}{RF_3} (|IS_1| (RF_1 - RF_3) + |IS_2| (RF_2 - RF_3)) \right\} ISize$
- $\vdots$
- IS_{c_d}: $\left\{ \sum_{k=1}^{c_d-1} |IS_k| + \frac{|IS_{c_d}|}{2} + \frac{1}{RF_{c_d}} \sum_{k=1}^{c_d-1} |IS_k| (RF_k - RF_{c_d}) \right\} ISize$

$\left\{ \sum_{k=1}^{c_d-1} |IS_k| + \frac{|IS_{c_d}|}{2} ISize \right\}$ 는 중복되는 인덱스 노드가 없을 때, 인덱스 집합 IS_{c_d}상에서 인덱스 노드를 얻는데 걸리는 평균 시간을 나타낸다. 본 논문의 방법은 중복된 인덱스 노드를 가지고 있으므로, 인덱스 집합 IS_{c_d}상에서 인덱스 노드를 얻는 데 걸리는 평균 시간은 $\left\{ \frac{1}{RF_{c_d}} \sum_{k=1}^{c_d-1} |IS_k| (RF_k - RF_{c_d}) \right\} ISize$ 만큼 평균이 증가해야 한다. 하나의 인덱스 채널을 통해서 인덱스 집합 c_d 상의 하나의 인덱스 노드를 얻는데 걸리는 평균 시간은 $ISize \sum_{j=1}^{c_d} \left[\left\{ \sum_{k=1}^{i-1} |IS_k| + \frac{|IS_j|}{2} + \frac{1}{RF_j} \sum_{k=1}^{i-1} |IS_k| (RF_k - RF_j) \right\} \frac{RF_j}{RF} \right]$ 가 된다. 여기서 c_i개의 인덱스 채널을 가지고 있으므로, 인덱스 노드를 얻는데 걸리는 평균 시간을 구하기 위해서 위의 수식을 c_i로 나누어야 한다. 그 결과, $L = \frac{ISize}{c_i} \sum_{j=1}^{c_d} \left[\left\{ \sum_{k=1}^{i-1} |IS_k| + \frac{|IS_j|}{2} + \frac{1}{RF_j} \sum_{k=1}^{i-1} |IS_k| (RF_k - RF_j) \right\} \frac{RF_j}{RF} \right]$ 의 값이 나온다. P와 L의 증명을 바탕으로 $IXT = P + L$임을 알 수 있다.

case 2 : $DT = \frac{DSize}{2} \sum_{j=1}^{c_d} \frac{RF_j}{RF} n_j$를 증명하자. DT는 인덱스 채널을 통해서 leaf 인덱스 노드를 받은 뒤에 데이타 채널 c_d를 통해서 데이타 노드를 얻는데 걸리는 평균 시간을 의미한다. 데이타 채널 DC_j를 통해서 전송되는 n_j개의 데이타 노드 중에서 하나의 데이타 노드를 접근할 확률이 $\frac{RF_j}{RF}$인 것은 자명하다. 데이타 채널 c_d를 통해서 데이타 노드들은 브로드캐스트 되므로, DT는 $\frac{DSize}{2} \left(\frac{RF_1}{RF} n_1 + \frac{RF_2}{RF} n_2 + \ldots + \frac{RF_{c_d}}{RF} n_{c_d} \right)$가 된다. 그러므로 $DT = \frac{DSize}{2} \sum_{j=1}^{c_d} \frac{RF_j}{RF} n_j$를 증명하였다.

case 1과 case2를 증명함으로써, c_i개의 인덱스 채널과 c_d개의 데이타 채널을 갖는 브로드캐스트 데이타의 평균 접근 시간 $AveAT$는 $IXT+DT$가 된다는 것을 보였다.

c개의 물리적 브로드캐스트 채널이 주어진 상태에서, 위에서 언급한 정리를 이용해서 인덱스 채널과 데이타 채널의 최적수를 결정할 수 있었다. 즉, 모든 가능한 c_i와 c_d의 조합을 시도함으로써 $AveAT$ 값들이 집합을 계산할 수 있다. 정리 5.1을 증명하기 위해서 우리는 먼저 표 2에서 언급한 동일한 값의 parameter를 사용해서 인덱스 채널과 데이타 채널을 이용한 가능한 모든 조합 [a: b]에 대해서 $AveAT$를 계산한다. 그 뒤, $AveAT$의 모든 순서쌍 [a: b]에 대하여 그림 16과 그림 17의 SIRAH의 값과 함께 그래프를 작성한다. 결과를 그래프로 작성한 그림 18과 그림 19는 정리 5.1의 효율성을 정확하게 증명하고 있다. 일례로, 10개의 채널이 있을 때, 그림 17은 SIRAH와 $AveAT$모두 데이타와 인덱스 채널의 수가 [8:2]일 때 최적의 성능을 보여주고 있다.

인덱스 채널과 데이타 채널의 최적 수는 부분적으로

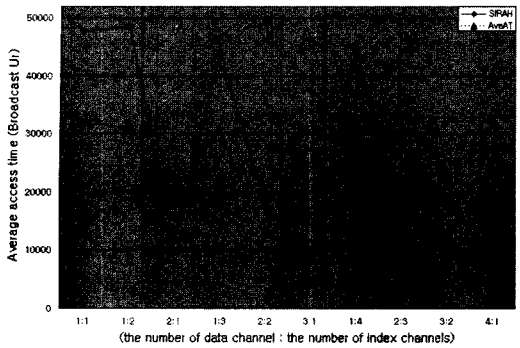


그림 18 채널 수 = 2, 3, 4, 5

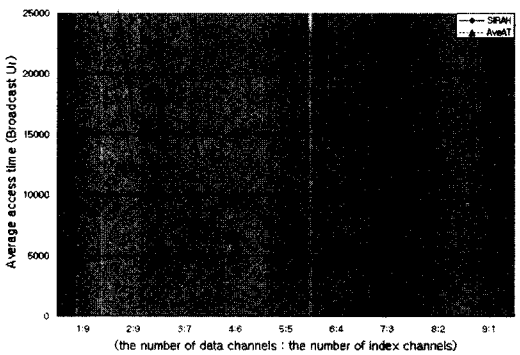


그림 19 채널 수 = 10

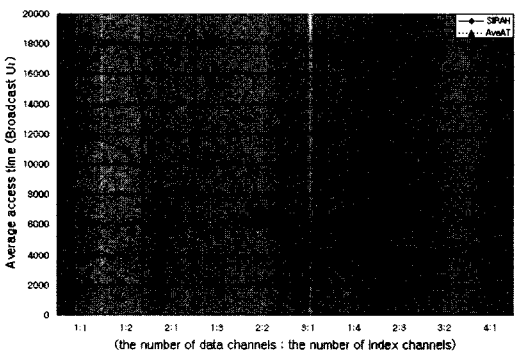


그림 20 채널 수 = 2, 3, 4, 5, R = 10

R(즉, 데이타 노드 크기와 인덱스 노드 크기 간의 비율)에 영향을 받음을 추측할 수 있다. 즉, *R*이 작으면 작을수록 더 많은 인덱스 채널을 필요로 하게 될 것이다. *R*값이 작다는 것은 인덱스 노드의 크기가 상대적으로 커진다는 것을 의미하므로, SIRAH방법에서 인덱스 트리를 브로드캐스트하기 위해서는 더 많은 인덱스 채널을 필요로 하게 될 것이다. 다음은 표 2의 parameter를 이용해서 $R=10$일 때, 앞에서 언급한 추측에 대해서 검증한다. simulation의 결과는 그림 20과

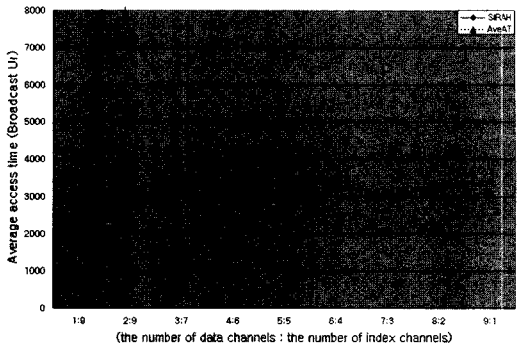


그림 21 채널 수 = 10, R=10

그림 21을 통해서 보이고 있다. 특히, 그림 21은 SIRAH와 *AveAT*모두 데이타 채널과 인덱스 채널의 개수가 [7:3]에서 최적의 성능을 보이고 있음을 명확하게 보이고 있다.

6. 결 론

본 논문에서는 이동 환경에서 멀티 브로드캐스트 채널을 통한 편향된 분포의 패턴을 갖는 브로드캐스트 데이타에 대한 효율적인 인덱스 할당 기법을 제안하였다. 이 방법은 온도가 높은 데이타와 인덱스 노드를 온도가 낮은 데이타와 인덱스 보다 빈번하게 브로드캐스트 함으로써 평균 접근 시간을 최소화한다. 또한, 인덱스 노드와 데이타 노트의 크기 차이로 인해서 채널의 대역폭 낭비를 줄임으로써 평균 접근 시간을 감소시켰다. 평균 접근 시간의 감소는 데이타 채널과 인덱스 채널을 각각 전용할 수 있도록 구분함으로써 가능하게 되었다.

Alphabetic 호프만 트리기반의 인덱스 할당 기법과 위상트리 기반의 인덱스 할당기법과 비교를 통해서 여러 가지 조건하에서 본 논문에서 제안한 기법의 성능을 분석하였다. 시뮬레이션 결과, 앞에서 언급한 두 가지 기법에 배해서 좋은 성능을 보였다. 다음으로, SIRAH 기법의 최적 성능을 얻기 위해서 물리적인 브로드캐스트 채널을 최적의 인덱스 채널수와 데이타 채널수로 구분하는 방법에 대해서 논의 하였다. 이를 위하여 본 논문에서는 인덱스와 데이타 채널의 최적 수를 결정하기 위한 효율적인 비용 모델을 개발하였다. 우리는 심도 있는 실험을 통해서 우리가 제안한 기법의 효율성을 입증하였다.

참 고 문 헌

[1] G. Forman and J.Zahorjan, "The Challenges of Mobile Computing," *In IEEE Computer*, 27(6), pp. 38~47, April 1994.

[2] T. Imielinski and B. Badrinath, "Wireless Mobile

Computing : Challenges in Data Management," *Comm. of ACM*, Vol 37, No. 10, pp. 18~28, 1994.

[3] S. Acharya, M. Franklin, S. Zdonik, and R. Alonso, "Broadcast Disks : Data Management for Asymmetric Communication Environment," *Proc. ACM SIGMOD International Conference. on Management of Data*, pp. 199~210, 1995.

[4] K.C.K Lee, Hong Va Leong, Antonio Si, "A semantic broadcast scheme for a mobile environment based on dynamic chunking," *Proc. of 20th International Conference on Distributed Computing Systems*, pp. 522~529, 2000.

[5] J. Juran, A.R. Hurson, N. Vijaykrishman and S. Boonsiriwattanakul, "Data Organization and Retrieval on Parallel Air Channels, Performance and Energy Issues," *Proceedings of the International Conference on HIgh Performance Computing (HiPC 2000)*, pp. 501~510, 2000.

[6] Ken C.K. Lee, Hong Va Leong, Antonio Si, "Semantic Data Access in an Asymmetric Mobile Environment," *Proceedings of the 3rd International Conference on Mobile Data Management(MDM'02)*, 2002.

[7] A.R. Hurson, Y.C. Chehadeh, and L.L. Miller, "Object organization on a single broadcast channel in a global information sharing environment," *Proc. of 24th Euromicro Conference*, Volume:2, pp. 1021~1028, 1998.

[8] W.C. Lee and D. L. Lee, "Using Signature Techniques for Information Filtering in Wireless and Mobile Environments," *Special Issue on Databases and Mobile Computing, Journal of Distributed and Parallel Databases*, Vol. 4, No. 3, July 1996, 205~227.

[9] T. Imielinski, S. Viswanathan, and B. R. Badrinath, "Data on Air: Organization and Access," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 9, No. 3, pp. 353~372, May/June 1997.

[10] N. Shivakumar, and S. Venkatasubramanian, "Efficient indexing for Broadcast Based Wireless Systems," *Mobile Networks and Applications*, pp.433~446, 1996.

[11] S. Lo and A. Chen, "Optimal Index and Data Allocation in Multiple Broadcast Channels," *Proceedings of 16th International Conference on Data Engineering*, 2000.

[12] S. Hameed and N. H. Vaidya, "Efficient Algorithms for Scheduling Single and Multiple Channel Data Broadcast," *Technical Report 97-002*, Department of Computer Science, Texas, A&M University, Feb. 1997.

[13] K. Tan and J. Yu, "An Analysis of Selective Tuning Schemes for Nonuniform Broadcast," *Data and Knowledge Engineering*, Vol. 22, No. 3, pp. 319~344, 1997.

[14] K. Prabhakara, K. Hua, and J. Oh, "Multi-Level Multi-Channel Air Cache Design for Broadcasting in a Mobile Environment," *Proceedings of 16th International Conference on Data Engineering*, 2000.

[15] D. Knuth, *"The Art of Computer Programming Second Edition*, Vol Ⅲ," Addison Wesley, 1998.

[16] Ni, W., S. Vrbsky, Q. Fang and J. Zhang, *Concurrency Control for Mobile Real-Time Databases Using Adaptive Broadcasting, The 20th IASTED International Conference on Applied Informatics*, pp. 425~430, Feb. 2002.

[17] H. Leong and A. Si, *"Data broadcasting strategies over multiple unreliable wireless channels," Proc of the Fourth International Conference on Information and Knowledge Management*, December, 1995

[18] H. D Schwetman, "CSIM: A C-based Process Oriented Simulation Language," *Proceeding of 1986 Winter Simulation Conference*, 1986.

[19] Mesquite Software, Inc, *CSIM18 Simulation Engine USER'S GUIDE*, 1987-1994.

[20] G.K Zipf. Human Behaviour and the Principle of Least Effort : An Introduction to Human Ecology. *Addison Wesley Press*, Cambridge, Massachusetts, 1949.

[21] J. Gray, P. Sundaresan, S. Englert, K. Baclawski, and P. Weinberger, "Quickly Generating Billion-record Synthetic Databases," *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, 1994.

[22] M. Chen, P. Yu, and K. Wu, "Indexed Sequential Data Broadcasting in Wireless Mobile Computing," *Proceedings of 17th ICDCS*, pp. 124~131, 1977.

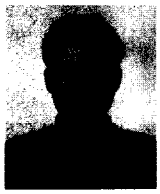
[23] M. Chen, K. Wu, and P. Yu, "Optimizing Index Allocation for Sequential Data Broadcasting in Wireless Mobile Computing," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 15, No. 1, Jan./Feb., 2003.

이 병 규

1994년 3월~2001년 2월 서강대학교 수학과 학사. 2001년 3월~2003년 2월 서강대학교 컴퓨터학과 석사. 2003년 2월~현재 삼성전자 무선사업부 연구원 관심분야는 WCDMA, GSM, 통신 protocol, 이동통신, 모바일컴퓨팅, 모바일데이타베이스

정 성 원

1988년 서강대학교 전자계산학 학사 1990년 M.S. in Computer Science at Michigan State Univ. 1995년 Ph.D. in Computer Science at Michigan State Univ. 1997년~2000년 한국전산원 선임연구원. 2000년~현재 서강대학교 컴퓨터학과 조교수. 관심분야는 Mobile Computing Systems, Mobile Databases, Spatial DB, Mobile Agents, Streaming Data Processing in Ubiquitous Computing Environments, Distributed Databases, ITS/GIS



이 승 중

1995년 3월~2002년 8월 서강대학교 컴퓨터학과 학사. 2002년 9월~현재 서강대학교 컴퓨터학과 석사과정. 관심분야는 이동통신, 모바일컴퓨팅, 모바일데이타베이스