# 이동컴퓨팅 환경에서 데이타의 접근빈도 및 시맨틱 관계를 고려한 방송 방법

## (Broadcast Method based on Data Access Frequencies and Semantic Relationships in Mobile Computing Environments)

최 성 환 † 정 성 원 †† 이 송 이 †††
(Sunghwan Choi) (Sungwon Jung) (Song-Yi Yi)

**요 약** 이동 컴퓨팅(mobile computing) 환경이 가지는 통신 대역의 협소함과 이동 기기의 에너지 제약 때문에 데이타 베이스 서버에서 다수의 이동 클라이언트로 데이타를 전달할 때는 브로드캐스트(broadcast)가 효과적이다. 기존의 여러 가지 브로드캐스트 방법은 클라이언트의 데이타 접근 빈도(access frequency)를 이용하여 전송 스케줄을 정하거나, 데이타들의 시맨틱 관계(semantic relationship)를 이용하여 전송 스케줄을 결정하였다. 데이타의 접근 빈도만을 반영하는 경우 클라이언트들이 접근하는 데이타들의 의미적 관계를 고려하지 않으므로 클라이언트가 밀접한 시맨틱 관계를 갖는 데이타를 차례로 접근해야 하는 경우 오랜 시간 동안 무선 채널을 듣고 있어야 한다. 시맨틱 관계만을 반영하여 전송 스케줄을 작성하면, 클라이언트가 시맨틱 관계는 없으나 접근 빈도가 높은 특정 데이타를 자주 접근할 필요가 있는 경우, 클라이언트의 데이타 접근 시간이 길어지게 된다. 이 논문에서는 데이타 접근 빈도와 시맨틱 관계를 함께 반영하여 이동 클라이언트의 데이타 접근 시간을 개선한 효율적인 하이브리드 데이타 브로드캐스트 방법을 제안한다. 우리가 제안하는 하이브리드 브로드캐스트 방법은 데이타 접근 빈도에 의해 브로드캐스트 스케줄을 생성한 후, 스케줄 상 데이타 전송 위치를 시맨틱 관계에 따라 조정한다. 시뮬레이션을 통해 기존의 방법들과 성능을 비교 분석하여 우리의 방법이 효율적임을 보인다.

**키워드** : 이동 컴퓨팅, 모바일 클라이언트, 브로드캐스트, 무선채널, 데이타 접근빈도, 시맨틱 관계

***Abstract*** Data broadcast is an effective data transmission method from a data base server to numerous mobile clients due to the restrictions on mobile environment such as low wireless communication bandwidth and energy shortage of mobile devices. There are various broadcast methods based on clients' data access frequencies or semantic relationship of data. The broadcast schedule based only on the access frequencies does not consider semantic relations of data, so that when a client needs to access a series of semantically related data, the client has to listen to the wireless channel for a long time. On the other hand, the broadcast schedule based only on semantic relationship of data makes data access time longer when clients highly request specific data which are not semantically related but frequently accessed. In this paper, we present an efficient data broadcast method based on not only data access frequencies but also semantic relationship to improve mobile clients' query response time. The new hybrid broadcast method we propose creates a data broadcast schedule according to the data access frequencies and then the schedule is adjusted to reflect semantic relationship of data. We show our method is efficient by experimental performance analysis.

**Key words** : mobile computing, mobile clients, broadcast, wireless channels, data access frequency, semantic relationships

† 비 회 원 : 서강대학교 컴퓨터학과
sunghwan@mclab.sogang.ac.kr
†† 종신회원 : 서강대학교 컴퓨터학과 교수
jungsung@ccs.sogang.ac.kr
††† 비 회 원 : 성신여자대학교 컴퓨터정보학부
yis@cs.sungshin.ac.kr
논문접수 : 2003년 1월 16일
심사완료 : 2003년 6월 25일

## 1. 서 론

무선통신 기술의 발달과 함께 고성능 휴대용 컴퓨터의 등장은 이동 컴퓨팅(mobile computing) 환경을 현실화 하였다. 이동 컴퓨팅 환경은 네트워크상의 사용자

들이 고정된 위치에 남아있기를 더 이상 요구하지 않으므로 사용자들의 자유로운 이동을 가능하게 한다. 자유로운 이동성은 이동 컴퓨팅 환경의 장점이지만 이로 인해 여러 가지 제약 조건이 발생한다. 이동 컴퓨터의 에너지 제한, 무선 통신 대역폭(wireless channel bandwidth)의 협소함, 무선 통신망과 기기의 안전성 결여, 단말기의 화면 크기 등이 그것이다[1,2].

위의 제약점을 가지는 이동 무선 컴퓨터 환경에서, 동시에 많은 수의 저 전력(low-powered) 컴퓨터나 PDA 등의 이동 클라이언트는 무선 통신 채널을 통해서 서버에게 데이타베이스 질의어 처리를 요구하고, 무선 통신 채널을 통해서 결과를 전달받는다. 이 때 많은 수의 이동 클라이언트의 질의를 한정된 대역폭과 자원을 갖는 값비싼 무선망 위에서 개별적으로 처리하는 대신 데이타 브로드캐스트 기법을 사용할 수 있다.

데이타 브로드캐스트는 이동 컴퓨팅 환경에서 불특정 다수의 클라이언트에게 저 비용으로 데이타를 전송하기 위한 방법이다. 브로드캐스트는 클라이언트가 서버에게 필요한 데이타를 따로 요구(request) 하지 않고, 그림 1과 같이 서버가 필요하다고 생각되는 공용 데이타(public data)를 클라이언트에게 주기적으로 전달한다. 개별 통신 기법에 비해 브로드캐스트의 가장 큰 장점은 이동 클라이언트의 수가 증가되어도 그에 따른 통신 비용의 부담이 전혀 생기지 않는다는 점이다. 브로드캐스트는 일정량의 채널을 다수의 사용자가 공유하여 사용하기 때문에 주파수의 효율성 측면에서 우수한 특징을 가진다. 또한 무선 통신의 특성상 데이타 수신은 데이타 송신보다 더 작은 에너지를 필요로 하기 때문에 브로드캐스트를 사용할 경우 이동 컴퓨터의 에너지 사용도 줄일 수 있다.

브로드캐스트 방법은 서버가 클라이언트의 데이타 요구를 예측하고, 그 예측 결과에 따라 데이타를 일방적으로 클라이언트에게 전송한다. 서버는 백 채널을 통해 들어오는 클라이언트의 요구 사항을 분석하거나, 클라이언트의 로그를 주기적으로 분석하여 클라이언트의 데이타 요구 형태를 예측한다. 이러한 서버의 예측이 정확할수록 필요한 데이타에 대한 접근 시간이 감소하여 클라이언트의 질의응답시간을 줄일 수 있다.

브로드캐스트 방법의 성능 평가는 두 가지 기준에 의해 이루어진다. 접근 시간(access time)과 튜닝 시간(tuning time)이다. 접근 시간은 클라이언트가 원하는 데이타를 얻기 위해, 브로드캐스트 채널을 듣기 시작해서 원하는 데이타를 모두 얻을 때까지의 시간을 나타낸다. 튜닝 시간은 클라이언트가 실제로 채널을 듣는 시간을 나타낸다. 클라이언트는 인덱스 정보 혹은 브로드캐스트 스케줄 정보를 이용하여 필요 없는 데이타가 브로드캐스트되는 시간동안에는 채널을 듣지 않는다. 즉, 클라이언트가 데이타를 수신하기 위해 에너지를 사용하는 시간을 나타낸다. 이러한 접근 시간과 튜닝 시간은 클라이언트에게 제공되는 브로드캐스트 서비스의 질을 나타내는 척도가 된다. 접근 시간과 튜닝 시간은 물리적인 시간(physical time) 단위(s, ms, ns...)나 논리적인 시간(logical time)을 측정 단위로 사용한다. 보통 논리적인 시간으로는 한 개의 데이타가 전송되는 시간을 기본 단위로 하여 사용하며, 브로드캐스트 유닛(broadcast unit)이라고 한다. 이 논문에서는 성능 평가를 위해 브로드캐스트 유닛 단위의 접근 시간을 사용한다.

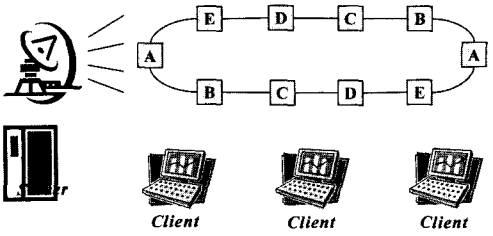


그림 1 데이타 브로드캐스트

데이타 브로드캐스트에서 접근 시간을 줄이기 위한 기존의 연구들은 크게 두 가지로 분류할 수 있다. 첫 번째는 클라이언트들이 특정 데이타를 다른 데이타들보다 자주 참조한다는 사실에 기반 하여, 클라이언트들의 데이타 참조 회수를 고려하여 데이타들의 전송 주기를 달리하는 접근 빈도 기반 브로드캐스트 방법이다[3,4]. 이 전송 방법은, 클라이언트 참조 회수가 높은 데이타는 서버에서 자주 전송하여 이러한 데이타는 이동 클라이언트가 짧은 시간 안에 접근하는 것이 가능하도록 한 것이다. 또 다른 연구 방향은 데이타 사이의 시맨틱 관계(semantic relationship)를 고려하여 시맨틱 관련성이 많은 데이타(클러스터)를 함께 전송하는 시맨틱 기반 브로드캐스트(semantic based broadcast)방법이다[6,8,10, 15]. 이 기법에서 데이타는 객체 지향 패러다임에 의해 하나의 객체(object)로 정의되고, 상속(inheritance), 집합(aggregation), 연합(association)과 같은 시맨틱 관계를 맺고 있다. 이 기법은 시맨틱 관계를 맺고 있는 데이타를 시간상 가깝게 브로드캐스트 함으로써 클라이언트가 필요한 데이타를 참조한 후 참조한 데이타와 시맨틱 관계가 있는 다음 데이타를 빠른 시간 안에 접근할 수 있도록 한 것이다.

이 논문에서는 이동 컴퓨팅 환경에서 데이타 접근 빈도와 시맨틱 관계를 모두 고려하여 데이타를 브로드캐스트 하는 방법을 제안한다. 데이타의 접근 빈도만을 가지고 브로드캐스트 스케줄을 생성하는 경우, 자주 참조되는 데이타의 접근 시간은 감소하나, 클라이언트의 질

의응답을 처리하기 위해서 필요한(매우 밀접한 시맨틱 관계를 맺고 있는) 관련 데이타의 접근 빈도가 낮은 경우에 이를 수신하기 위해 클라이언트가 상대적으로 오랜 시간 동안 무선 채널을 듣고 있어야 한다. 데이타의 시맨틱 관계만을 고려하여 전송할 경우에는 데이타의 접근 빈도를 고려하지 않고 모든 데이타를 동일한 시간 간격으로 동일한 빈도로 브로드캐스트 하므로 이동 클라이언트들이 접근 빈도가 높은 데이타를 집중적으로 참조하는 경우 필요한 데이타를 얻기 위해 클라이언트가 대기하는 시간이 길어진다. 우리가 제안하는 브로드캐스트 기법은 데이타 접근 빈도와 시맨틱 관계, 두 가지 모두를 고려하여 데이타를 브로드캐스트 함으로써 접근 빈도가 높은 데이타는 좀 더 빈번하게 브로드캐스트 하면서, 동시에 시맨틱 관계가 있는 데이타를 함께 전송하여 이동 클라이언트의 데이타 접근 시간이 개선되도록 하였다.

이 논문의 구성은 다음과 같다. 먼저 2장에서는 기본적인 데이타 브로드캐스트 기법과 기존의 연구들에 대해 알아본다. 3장에서는 기존 연구의 문제점을 지적하고, 새로운 데이타 브로드캐스트 방법을 제안하며, 4장에서는 실험을 통해 제안된 방법의 성능을 기존의 방법들과 비교 분석하여 평가한다. 마지막으로 5장에서는 결론과 향후 연구 방향을 기술한다.

## 2. 관련 연구

이 논문에서 가정하는 브로드캐스트 환경은 하나의 서버가 무선 통신망을 이용하여 데이타를 방송하고 이를 불특정 다수의 클라이언트들이 수신하는 것이다. 서버는 일정한 데이타 집합을 주기적으로 브로드캐스트 하여 클라이언트들이 질의처리를 하는데 필요한 데이타를 수신할 수 있도록 한다. 클라이언트는 질의를 처리하기 위해 먼저 클라이언트의 캐쉬(cache)에 필요한 데이타가 있는지 검색한다. 만약 필요한 데이타가 캐쉬에 있다면 그것을 이용하여 질의를 처리하지만 만약, 캐쉬에 필요한 정보가 없다면 현재 서버로부터 브로드캐스트 되는 데이타들을 검색하여 필요한 정보를 수신한다.

기본적인 데이타 브로드캐스트 방법은 서버에 있는 데이타들을 있는 그대로 주기적으로 전송한다. 클라이언트는 이러한 방법으로 서버로부터 전송되는 데이타들 중에서 클라이언트 자신의 질의를 처리하는데 필요한 데이타를 선별하여 수신하고, 질의를 처리하여 그 결과를 사용자에게 보여준다. 이러한 브로드캐스트 스케줄 방법은 모든 데이타들에 대한 접근 시간이 동일하다. 즉, 클라이언트들이 필요한 데이타를 수신하기 위해 기다려야 하는 평균 시간이 모든 데이타들에 대해 같다. 그림 2에서 클라이언트가 데이타 a를 받기 위해 기다려야 하는 접근 시간과 데이타 b를 받기 위해 기다려야 하는 접근 시간은 평균적으로 동일하다. 이러한 데이타 브로드캐스트 방법은, 클라이언트의 데이타에 대한 상대적인 접근 빈도는 전혀 고려되지 않았다는 것을 뜻한다.

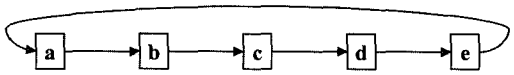


그림 2 기본적인 데이타 브로드캐스트 방법

하지만 클라이언트들의 데이타 접근 형태는 모든 데이타들에 대해 동일하지 않다. 즉, 클라이언트들은 데이타를 참조하는데 있어서 모든 데이타를 균일한 비율로 참조하지 않는다. 전체 데이타들 중에서 특정 데이타를 더 많은 회수로 참조하거나, 반대로 전체 데이타 중에서 일부분은 거의 참조하지 않는다. 혹은 특정 데이타는 다른 데이타에 비해 빨리 접근하는 것이 매우 중요할 수 있다. 예를 들어, 위의 그림 2와 같이 브로드캐스트가 이루어지고 있는 상황에서 만약 클라이언트들이 데이타 b를 중요하게 생각해서, 빠른 시간 안에 b를 얻기를 원한다고 해도, 클라이언트들이 데이타 b에 접근하기 위해서 기다려야 하는 접근 시간은 상대적으로 덜 중요한 다른 데이타 a, c ,d, e를 얻기 위해 기다려야하는 접근 시간과 동일하다. 또 다른 관점에서 볼 때, 클라이언트가 질의처리를 위해 하나의 데이타가 아닌 시맨틱 관계가 있는 여러 개의 데이타를 필요로 하는 경우, 예를 들어, 데이타 e가 데이타 a에서 파생되어 대부분의 클라이언트들이 데이타 a를 참조한 후 데이타 e를 참조한다고 하면, 이 들 클라이언트는 데이타 a를 참조한 후 데이타 e에 접근하기 위해 전체 브로드캐스트 스케줄의 처음부터 끝까지 수신하고 있어야 한다.

데이타를 있는 그대로 전송하는 브로드캐스트 방법은 클라이언트들의 다양한 데이타 접근 요구를 만족시켜주지 못하므로, 이를 개선하기 위해 다양한 브로드캐스트 기법이 연구되고 있다[3,4,6,8,9,10,15]. 이 들 기법은 크게 두 가지로 나뉠 수 있는데 하나는 데이타의 중요도나 접근 빈도에 기반한 것이고, 다른 하나는 데이타의 시맨틱 관계에 기반한 것이다.

### 2.1 브로드캐스트 디스크

이 절에서는 데이타의 상대적인 중요성이나 데이타들마다 서로 다른 클라이언트 데이타 참조 회수 혹은 데이타의 접근 빈도 차이를 고려한 접근 빈도 기반 브로드캐스트 방법을 설명한다. Acharya는 데이타들의 전송 빈도를 달리하여, 클라이언트들이 데이타를 얻기 위해 기다려야 하는 시간을 차별하는 브로드캐스트 방법 - 브로드캐스트 디스크(Broadcast disk)를 제안했다[3]. 이 기법은 클라이언트들이 자주 참조하는 데이타들은


그림 3 데이타의 접근 빈도에 기반한 브로드캐스트 스케줄

다른 데이타들보다 전송회수를 크게 하고, 그렇지 않은 데이타들은 전송회수를 적게 한다. 그림 3은 데이타의 접근 빈도에 따라 데이타의 전송 회수를 다르게 한 브로드캐스트 방법을 나타낸다. 데이타 a의 접근 빈도가 다른 데이타들 b, c의 접근 빈도에 비해 2배인 경우, 전체 브로드캐스트 스케줄에서 데이타 a의 전송 회수를 데이타 b, c의 전송 회수의 두 배가 되도록 한다. 그림 3에서 보이는 것과 같이 두 가지 형태로 데이타를 전송할 수 있다.

우선 그림 3-I은 데이타의 중요도가 높은 a를 먼저 전송한 후에 b, c를 전송하는 방식이다. 하지만 이와 같이 데이타를 브로드캐스트 하면, 클라이언트들이 데이타 a에 접근하기 위해 기다려야 하는 시간이 일정하지가 않다. 이렇게 데이타를 브로드캐스트 하는 것은 특정 데이타(a)가 다른 데이타들(b, c) 보다 자주 전송함으로써 a의 중요도에 따른 접근 빈도는 만족시키지만, 클라이언트들의 데이타 a의 평균 접근 시간은 규칙적이지 않다. 클라이언트가 데이타 a를 참조하기 위해 기다려야 하는 접근 시간은 최소 1 브로드캐스트 단위(broadcast unit)이며(첫 번째 나타난 a를 참조한 후에 다시 a를 참조하는 경우), 최대 3 브로드캐스트 단위이다(두 번째 나타난 a를 참조한 후에 다시 a에 참조하는 경우). 이것은 데이타 a의 전송 간격이 일정하지 않기 때문에 발생하는 문제이다.

데이타를 일정한 간격으로 전송하기 위해서, 그림 3-II와 같이 전송 빈도가 다른 데이타들을 서로 교차시켜 브로드캐스트 하도록 한다. 그림 3-II의 경우 클라이언트가 a를 받기 위해 기다려야 하는 접근 시간은 최대 2이며, a의 전송 간격도 일정하다. 접근 빈도 기반 브로드캐스트 방법에서는 데이타들의 상대적인 접근 빈도에 따라 데이타의 전송 회수가 결정된다. 우선, 접근 빈도에 따라 데이타들을 분류하여, 접근 빈도가 비슷한 데이타들을 하나의 그룹(group)으로 만들고, 이 그룹들을 회전 속도가 다른 디스크(disk)로 간주한다. 각 디스크들의 회전 속도가 다르기 때문에 그룹에 속한 데이타들의 전송 회수는 다르게 된다.

그림 4는 이러한 방법으로 데이타의 전송 회수를 다르게 하면서, 데이타들의 평균 접근시간을 균일하게 만드는, 데이타 접근 빈도 기반 브로드캐스트 스케줄 생성 알고리즘이다. 이 알고리즘에서 뜨거운 데이타(hot data)와 차가운 데이타(cold data)는 클라이언트의 데이타 접근 빈도(혹은 중요도)에 의해 결정된다. 온도에 따라 정렬된 데이타(페이지)는 유사한 접근 빈도를 가진 데이타끼리 묶음(디스크)을 지어 상대 접근 빈도를 부여

1. Order the pages from hottest (most popular) to coldest
2. Partition the list of pages into multiple ranges, where each range contains pages with similar access probabilities.
   These ranges are referred to as disks
3. Choose the relative frequency of broadcast for each of the disks
4. Split each disk into a number of smaller units.
   *These units are called chunks.* ($C_{ij}$ *refers to the* $j^{th}$ *chunk in disk i)*
   Max_chunk = the Least Common Multiple(LCM) of the relative frequencies
   Split each disk i into num_chunks(i) = max_chunk / rel_freq(i) *chunks*
5. Create the broadcast program by interleaving the chunks of each disk in the following manner :

```
01  for i := 0 to max_chunks - 1
02      for j := 1 to num_disks
03          Broadcast chunk C_{j,(i mod num_chunks(j))}
04      end for
05  end for
```

그림 4 데이타 접근 빈도 기반 브로드캐스트 스케줄 작성 알고리즘[3]

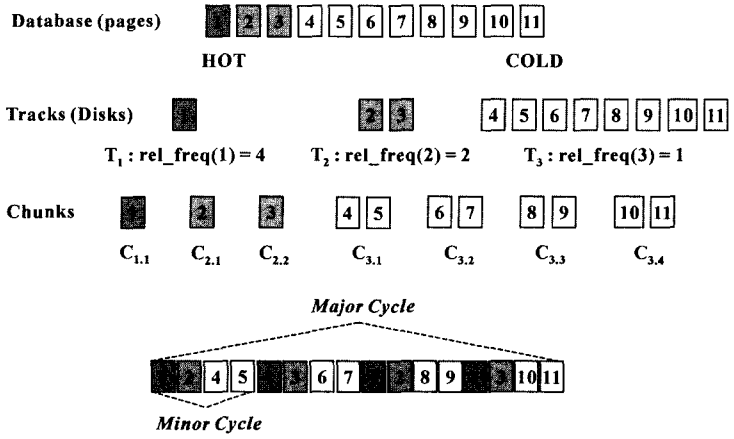



그림 5 데이타의 접근 빈도를 반영한 브로드캐스트 스케줄[3]

받는다. 각 묶음은 일정 크기의 청크(chunk)를 부여받는데, 뜨거운 데이타(접근 빈도가 높은 디스크)는 청크의 수가 적다. 디스크의 청크를 돌아가며 브로드캐스트하면 적은 수의 청크를 갖는 디스크는 많은 수의 디스크 사이에 반복되어 브로드캐스트 된다. 자세한 설명은 [3]에 기술되어 있다. 그림 4의 알고리즘을 이용하여 생성한 브로드캐스트 스케줄은 그림 5와 같으며, 그림 5의 브로드캐스트 스케줄은 네 개의 마이너 사이클(minor cycle)로 구성된 하나의 메이저 사이클(major cycle)의 형태이다. 메이저 사이클은 서버에 있는 모든 데이타가 브로드캐스트 된 주기를 의미한다.

### 2.2 데이타의 시맨틱 관계에 기반한 브로드캐스트 방법

클라이언트가 질의처리를 할 때 한 개의 데이타만을 필요로 하는 경우도 있지만 여러 개의 데이타를 필요로 할 때도 있다. 클라이언트가 질의처리를 위해 필요로 하는 이 데이타들은 서로 어떠한 관계를 가지고 있으며, 이러한 데이타들의 관계를 시맨틱 관계라고 한다. 데이타의 시맨틱 관계를 고려한 브로드캐스트 방법이 연구되어 왔으며, Hurson은 시맨틱 관계에 따라 브로드캐스트 되는 데이타들의 전송 순서를 결정하는 방법을 제안했다[10].

[10]에서는 데이타를 하나의 객체로 취급하여 데이타들의 시맨틱 관계를 파악하였다. 데이타를 하나의 객체로 간주한 객체 지향 관점(object-oriented paradigm)

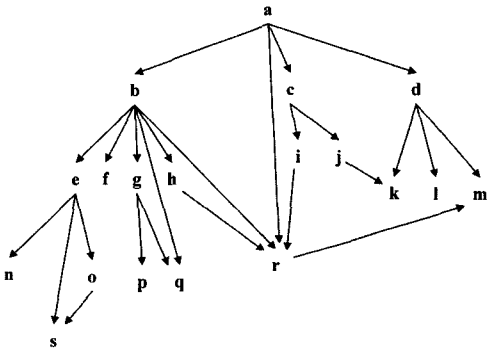



그림 6 데이타의 시맨틱 관계를 나타낸 그래프

에서 데이타들의 시맨틱 관계를 파악하였고, 그렇게 파악된 시맨틱 관계를 DAG (Direct Acyclic Graph)를 이용하여 나타내었다. DAG를 이용한 데이타들의 시맨틱 관계 표현법에서 노드(node)는 객체-데이타를 의미하고 간선(edge)은 객체사이의 연결을 의미한다.

시맨틱 관계를 나타낸 그래프(graph)를 브로드캐스트하기 위해서는 그래프에 나타난 데이타들의 관계를 선형순서(linear order)로 나타내야 한다. 그래프를 선형순서로 나타내는 DFS(Depth First Search), BFS (Breadth First Search)등 다양한 방법들이 있으며, 표 1은 시맨틱 관계를 표현한 그림 6의 DAG를 DFS와 BFS를 이용하여 선형순서로 나타낸 것이다.

표 2는 그림 7의 그래프에서 생성 가능한 선형순서와

표 1 DFS와 BFS에 의한 선형순서

| Clustering Method | Resulting Sequence |
|---|---|
| Depth First | *a b e n s o f g p q h r m c i j d k l* |
| Breadth First | *a b r c d e f g q h i j k l m n s o p* |

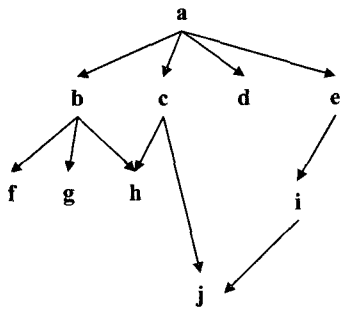



그림 7 데이타의 시맨틱 관계를 나타낸 그래프

비용(cost)을 나타내고 있다. 데이타들 간의 관계를 나타내는 간선의 가중치(weight)는 다양한 값을 가질 수 있지만, 그림 7에서는 모두 1로 한다. 따라서 표 2에서 비용은 관계가 있는 데이타들의 거리를 의미한다. 개별 비용(individual cost)은 관계가 있는 데이타(간선으로 직접 연결된 데이타)들의 선형순서 상에서의 거리를 의미하며, 총 비용(total cost)은 이들 개별 비용을 모두 합한 것이다.

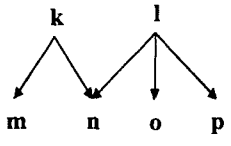
클라이언트가 필요한 데이타를 얻기 위해 기다려야 하는 시간을 최소화하기 위해서는 시맨틱 관계가 있는 데이타들의 거리, 즉 비용이 작게 나타난 선형순서로 데이타를 브로드캐스트 해야 한다. 모든 클라이언트에 대한 데이타 브로드캐스트 효과 즉, 총 비용을 생각해야 하기 때문에 총 비용이 가장 작은 것을 찾아내야 한다. 그림 7에서 총 비용이 가장 적은 선형순서를 찾아내기 위해 가능한 모든 데이타 전송 순서를 나열한 후에 총 비용이 가장 적은 것을 검색하는 방법을 사용하고 있다. 하지만 이 방법은 객체의 개수가 작으면 괜찮지만, 개수가 커지면 실행비용이 기하급수적으로 늘어나게 되므로 실제로 사용하기에는 무리가 있다.

따라서 [10]에서는 합당한 비용(reasonable cost)을 갖는 선형순서를 생성하는 휴리스틱 법칙(heuristic rule)을 제안하고 있다. 그림 8은 휴리스틱 법칙을 이용하여 선형순서를 생성하는 예이고, 그림 9는 휴리스틱 법칙을 이용하여 선형순서를 생성하는 알고리즘이다. 이 알고리즘은 기본적으로 DFS를 사용하여 시맨틱 그래프에서 선형순서를 추출하는데, 휴리스틱 규칙을 사용해서 부모가 모두 방문된 노드 중에서 가장 작은 수의 자식 노드를 먼저 선택한다. 그러나, 이들의 알고리즘은 기본적으로 DFS를 사용하여 데이타 접근의 선후 관계를 보장하지 못하고, 시맨틱 비용 면에서 더 개선할 여지가 있다. 알고리즘의 자세한 설명은 [10]에 있다.

그림 9에 설명되어 있는 알고리즘을 그림 7의 그래프에 대해 실행시키면 표 2의 5번째 혹은 11번째 결과와 같은 선형순서를 생성한다. 이것은 c 혹은 d 에서 어느 것을 먼저 선택하느냐에 따라 결정된다. 결과에서 알 수 있듯이 이 알고리즘은 최적의 선형순서를 생성하지는 않지만 다항식 시간(polynomial time) 안에 합당한 비용(reasonable cost)의 결과를 얻을 수 있는 장점을

표 2 가능한 선형순서 및 비용

| # | 선형순서 | 개별 비용 | | | | | | | | | | | 총 비용 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | ab | ac | ad | ae | bf | bg | bh | ch | cj | ei | ij | |
| 1 | abfgchdeij | 1 | 4 | 6 | 7 | 1 | 2 | 4 | 1 | 5 | 1 | 1 | 33 |
| 2 | abfgcheijd | 1 | 4 | 9 | 6 | 1 | 2 | 3 | 1 | 4 | 1 | 1 | 34 |
| 3 | abcdefghij | 1 | 2 | 3 | 4 | 4 | 5 | 6 | 5 | 7 | 4 | 1 | 42 |
| 4 | abgfeichjd | 1 | 6 | 9 | 4 | 2 | 1 | 6 | 1 | 2 | 1 | 3 | 36 |
| 5 | acdeijbhgf | 6 | 1 | 2 | 3 | 3 | 2 | 1 | 6 | 4 | 1 | 1 | 30 |
| 6 | **adeicjbhgf** | 6 | 4 | 1 | 2 | 3 | 2 | 1 | 3 | 1 | 1 | 2 | **26** |
| 7 | adecbihgfj | 4 | 3 | 1 | 2 | 4 | 3 | 2 | 3 | 6 | 3 | 4 | 35 |
| 8 | adecbhgfij | 4 | 3 | 1 | 2 | 3 | 2 | 1 | 2 | 6 | 6 | 1 | 31 |
| 9 | adecijbhgf | 6 | 3 | 1 | 2 | 3 | 2 | 1 | 4 | 2 | 2 | 1 | 27 |
| 10 | adbgfcheij | 2 | 5 | 1 | 7 | 1 | 2 | 4 | 1 | 4 | 1 | 1 | 29 |
| 11 | adceijbhgf | 6 | 2 | 1 | 3 | 3 | 2 | 1 | 5 | 3 | 1 | 1 | 28 |




| Linear Sequence | Individual Costs | | | | | Total Cost |
|---|---|---|---|---|---|---|
| | km | kn | ln | lo | lp | |
| kmlnop | 1 | 3 | 1 | 2 | 3 | 10 |
| kmlonp | 1 | 4 | 2 | 1 | 3 | 11 |
| kmlpon | 1 | 5 | 3 | 2 | 1 | 12 |

그림 8 휴리스틱 법칙을 이용하여 생성한 선형순서[10]

```
APPROXIMATE LINEAR ORDER
 1  traverse dag using DFS traversal
 2     as each node N is traversed
 3     append N to sequence
 4     remove N from { nodes to be traversed}
 5     NextNode ← child, using the rules
 6     if { non-free children having all their parents already
          in the sequence } ≠ ∅
 7        Set ← { non-free children having all their parents already
                  in the sequence }
 8     else
 9        if { free children } ≠ ∅
10           Set ← { free children }
11     NextNode ← child ∈ Set | child with least # of descendants
```

그림 9 휴리스틱 법칙을 이용하여 선형순서를 생성하는 알고리즘[10]

지니고 있다.

## 3. 하이브리드 브로드캐스트 방법

이 장에서는 이동 컴퓨팅 환경에서 데이타 접근 빈도와 시맨틱 관계를 모두 고려하여 데이타 브로드캐스트 스케줄을 작성하는 방법을 제안한다. 데이타의 접근 빈도만을 가지고 브로드캐스트 스케줄을 생성하는 경우, 자주 참조되는 데이타의 접근 시간은 감소하나, 클라이언트의 질의응답을 처리하기 위해서 필요한 관련 데이타(시맨틱 관계가 있는 데이타)의 접근 빈도가 낮은 경우에는, 이를 수신하기 위해 클라이언트가 상대적으로 오랜 시간 동안 무선 채널을 듣고 있어야 한다. 따라서 클라이언트의 접근 시간이 길어지게 되는 문제점을 지니고 있다. 반대로, 데이타의 시맨틱 관계만을 고려하여 전송할 경우에는 데이타의 접근 빈도를 고려하지 않고 모든 데이타를 동일한 시간 간격으로 동일한 빈도로 브로드캐스트 하므로 이동 클라이언트들이 접근 빈도가 높은 데이타를 집중적으로 참조하는 경우 필요한 데이타를 얻기 위해 클라이언트가 대기하는 시간이 길어진다.

제안하는 브로드캐스트 기법은 데이타 접근 빈도와 시맨틱 관계, 두 가지 모두를 고려하여 데이타 브로드캐스트 스케줄을 작성하는 방법이다. 이러한 방법으로 데이타를 브로드캐스트 함으로써 접근 빈도가 높은 데이타는 좀 더 빈번하게 전송되면서, 동시에 시맨틱 관계가 있는 데이타가 가까운 시간 안에 전송되어 이동 클라이언트의 데이타 접근 시간이 개선되도록 하였다. 우리가 제안하는 새로운 브로드캐스트 알고리즘을 설명하기 위해 필요한 사항들을 3.1절과 3.2절에서 살펴본다.

### 3.1 시맨틱 관계

앞서 기술한 바와 같이 이 논문은 데이타의 접근 빈도와 함께 데이타들의 시맨틱 관계도 반영한 브로드캐스트 방법을 제안한다. 데이타들의 시맨틱 관계는 2.2절에서와 같이 DAG를 이용하여 나타낸다. 그래프 간선의 가중치는 그 값이 클수록 밀접하게 관련되어 있거나 데이타를 함께 참조하는 비율이 높다는 것을 표시하며, 따라서 높은 가중치를 갖는 간선으로 연결된 데이타는 브로드캐스트 스케줄 상에서 데이타 간 거리가 짧아야 한다.

그림 10은 이 장에서 제시하는 알고리즘을 설명하기 위한 예제의 데이타 접근 빈도와 시맨틱 관계 그래프를 보인 것이다. 접근 빈도가 가장 높은 a와 중간 수준의 b, c 낮은 수준의 d에서 j까지 10개의 데이타가 있으므로 이들은 접근 빈도에 따라 3개의 디스크로 분류되었다. 간선의 가중치가 모두 같다고 가정하여 가중치 값을 표시하지 않았다.

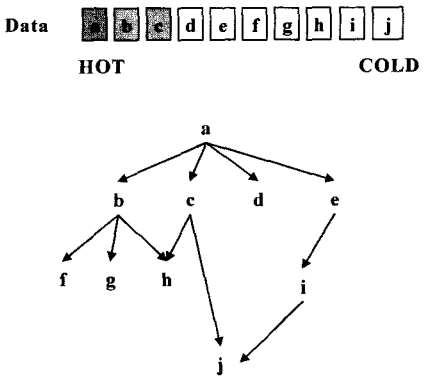



그림 10 데이타의 접근 빈도에 따른 분류와 시맨틱 관계 그래프(DAG)

### 3.2 선형순서

DAG로 표현된 시맨틱 관계를 선형순서로 나열하기 위해 여러 가지 방법을 사용할 수 있다. 클라이언트들이 루트(root)로부터 데이타들을 차례로 접근해나가는 과정을 반영하기 위해 그림 11과 같이 DFS나 BFS 탐색 방법을 사용하여 데이타를 순차적으로 나열할 수 있다.

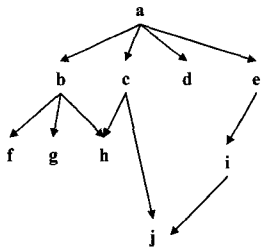
DFS나 BFS는 비교적 쉽게 데이타를 순차적으로 나열하지만, 종속적인 시맨틱 관계가 있는 데이타간의 선행 관계는 반영하지 못한다. 그림 11에서 데이타 j는 c와의 관계도 있지만 i와의 관계도 지니고 있다. 이런 경우에는 j는 c와 i 다음에 나타나는 것이 올바른 것이다. 이러한 선행 관계를 반영하여 스케줄하기 위해 위상 정렬(Topological Sort - 이하 TS)을 사용한다. TS를 사용하여 선형순서로 나타내면 화살표로 표현된 간선의 선·후 관계를 반드시 지키기 때문이다. BFS나 DFS 혹은 TS를 이용한 방법 중에서 어느 것이 더 좋은 것인지는 클라이언트의 데이타 접근 형태와 시맨틱 관계의 유사성에 따라 결정된다. 즉, 클라이언트의 데이타 접근 형태가 TS 형태와 유사하다면 TS를 이용한 방법이 더 좋은 성능을 나타낼 것이다. 이 논문에서는 BFS와 DFS, TS 모두를 성능 평가 부분에서 비교하였다.

그림 12는 우리가 고안한 위상 정렬 알고리즘 *Modified TS*이다. 이 알고리즘의 보다 자세한 설명과 분석은 [18]에 있다. 그림 12와 같이 위상 정렬 알고리즘을 수정한 이유는 시맨틱 관계에 있는 노드들을 가까운 시간 안에 선택하여 개별 비용의 합을 최소화하기 위해서이다. Modified TS는 우선 기본적으로 위상정렬을 사용하므로 데이타 접근의 선후관계에 일치하는 브로드캐스트 스케줄을 생성한다. 또한 작은 수의 자식을 갖는 노드를 먼저 선택하고, 자식수가 같을 때는 부모의 수가 많은 노드를 먼저 선택함으로써 시맨틱 비용 면에서 BFS나 DFS보다 작은 개별비용을 보장한다[18]. 표 3은 각 선형순서 생성 방법을 이용하여 나온 선형순서와 그 비용을 계산한 것이다. 단 간선의 가중치는 모두 1로 하였으며, 비용은 2.2절에서와 마찬가지로 시맨틱 관계가 있는 데이타들의 거리를 의미한다. 이 논문의 모든 실험에서 위상 정렬은 그림 12의 *Modified TS*를 사용하였다.

하지만, 실제 시맨틱 관계를 나타내는 그래프에서 간선의 가중치는 위의 상황과 같이 모두 1이 아니라 다양한 값을 가진다. 이러한 경우 가중치가 높은 간선의 노드를 먼저 방문하도록 우선 순위를 둔다. Modified TS의 경우 그림 14와 같은 방법으로 가중치에 대한 우선순위를 두도록 하며, 그림 15의 예제는 가중치를 갖는 시맨틱 그래프와 위상 정렬 결과를 보인 것이다. DFS,

1) 부모가 없는(또는 모두 제거된) 노드를 고른다.
2) 1)의 조건에 해당하는 노드가 여러 개인 경우는 그 중 자식 수(outgoing edge)가 가장 작은 노드를 선택한다.
3) 자식수가 같을 때는 제거된 부모의 수(incoming edge)가 가장 많은 노드를 선택한다.
4) 선택된 노드를 출력하고 그래프에서 삭제한다. 그래프 노드 집합이 공집합이 될 때까지 1)번의 규칙부터 적용하여 차례로 모든 노드를 삭제한다.

그림 12 수정된 위상 정렬 알고리즘 (*Modified TS*)




DFS : *a b f g h c j d e i*

BFS : *a b c d e f g h j i*

그림 11 그래프와 선형순서

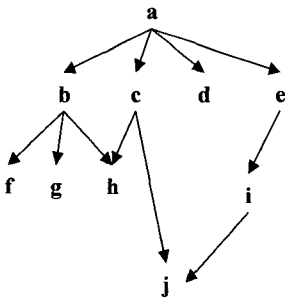



그림 13 데이타의 시맨틱 관계를 나타낸 그래프

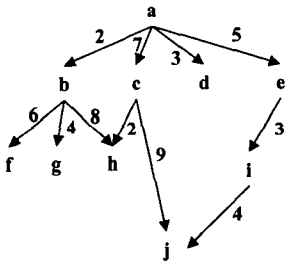
표 3 다양한 선형순서 및 비용

| 노드선택 | 선형순서 | 개별 비용 | | | | | | | | | | | 총 비용 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | ab | ac | ad | ae | bf | bg | bh | ch | cj | ei | ij | |
| **Optimal Schedule** | *adeicjbhgf* | 6 | 4 | 1 | 2 | 3 | 2 | 1 | 3 | 1 | 1 | 2 | 26 |
| **DFS** | *abfghcjdei* | 1 | 5 | 7 | 8 | 1 | 2 | 3 | 9 | 1 | 1 | 7 | 45 |
| **BFS** | *abcdefghji* | 1 | 2 | 3 | 4 | 4 | 5 | 6 | 5 | 6 | 5 | 9 | 50 |
| **TS** | *abcdefghij* | 1 | 2 | 3 | 4 | 4 | 5 | 6 | 5 | 8 | 4 | 2 | .49 |
| ***Modified TS*** | *adeicjbhfg* | 6 | 4 | 1 | 2 | 2 | 3 | 1 | 3 | 1 | 1 | 2 | 26 |

BFS 방법도 마찬가지로 가중치가 높은 것을 먼저 방문하는 방식으로 가중치를 고려하여 선형순서를 생성할 수 있다. 4장의 실험에서는 그림 15와 같이 간선에 가중치를 가진 시맨틱 그래프를 대상으로 하지만, 우리가 제안하는 하이브리드 브로드캐스트 스케줄링 알고리즘을 설명하기 위한 예제에서는 간선의 가중치를 1로 한다.

1. 부모가 없는(또는 모두 제거된) 노드를 고른다.
2. 간선의 가중치가 높은 노드를 선택한다.
3. 2)의 조건에 해당하는 노드가 여러 개인 경우는 그 중 자식 수(outgoing edge)가 가장 작은 노드를 선택한다.
4. 자식수가 같을 때는 제거된 부모의 수(incoming edge)가 가장 많은 노드를 선택한다.
5. 선택된 노드를 출력하고 그래프에서 삭제한다. 그래프 노드 집합이 공집합이 될 때까지 1)번의 규칙부터 적용하여 차례로 모든 노드를 삭제한다.

그림 14 가중치를 고려하는 *Modified TS*




***생성된 선형순서 : a c e d i j b h f g***

그림 15 가중치 고려 Modified TS

### 3.3 하이브리드 브로드캐스트 알고리즘

다음은 우리가 제안하는 새로운 브로드캐스트 알고리즘의 변수이다.

$1 \le j \le num\_disk$, $num_disk$는 디스크의 개수 일때,

- *disk(j)* : 접근 빈도가 같은 데이타의 집합
- *rel_freq(j)* : 디스크 *j*의 상대 접근 빈도(relative access frequency).

항상 정수 값을 갖는다.

예) *rel_freq(6) = 1, rel_freq(3) = 4*

6번 디스크의 상대적 접근 빈도가 1임을 의미하고, 3번 디스크의 상대적 접근 빈도는 6번 디스크의 4배임을 나타낸다.

- *m* : 전체 브로드캐스트 스케줄을 구성하는 마이너 사이클의 개수. 이것은 다음과 같이 계산한다.

*m = rel_freq(j)의 최소공배수*, LCM{*rel_freq(j)*}

브로드캐스트 디스크 방법에 의한 브로드캐스트 스케줄의 메이저 사이클은 여러 개의 마이너 사이클로 구성된다.

- bin : 데이타의 접근 빈도에 기반 하여 분류된 디스크들의 전송 주기는 마이너 사이클을 단위로 하여 표현할 수 있다. 메이저 사이클이 *m*개의 마이너 사이클로 구성되어 있다면 *disk(j)*의 전송 주기는 *m/rel_freq(j)* 이며 이 값을 디스크에 속한 데이타들을 담는 *bin(j)*으로 한다.

그림 16에서 디스크 b는 전송 주기가 2개의 마이너 사이클이며, b가 속하는 빈(bin)의 크기는 2개의 마이너 사이클을 합한 것이다. 이러한 방법으로 각 디스크 별로 배정되는 빈(bin)의 크기가 결정되면, 디스크들은 해당 빈(bin)에 디스크에 속한 데이타들을 할당한다. 단 빈(bin)에 데이타들을 할당할 때 데이타들의 시맨틱 관계를 고려해야 하므로, *Modified TS*를 이용하여 얻은 선형 순서를 참조하여, 선형 순서 상에 나타난 데이타들의 순서를 지키면서 데이타들을 빈(bin)에 넣는다. 그림 18은 그림 13의 그래프와 *Modified TS*에 의해 생성된 선형순서를 고려하여 빈(bin)에 각 디스크들의 데이타들을 할당한 것을 나타낸 것이다. 디스크들의 상대적인 접근 빈도는 예시를 위해 4 : 2 : 1로 하였다.

이제 이렇게 데이타들을 할당한 빈(bin)들을 종합하여 브로드캐스트 스케줄을 작성한다. 빈(bin)에 할당된 데이타들은 현재 그 빈(bin)을 구성하고 있는 마이너 사

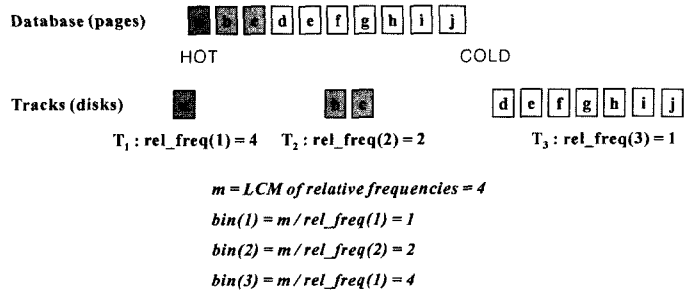



그림 16 각 디스크들의 빈(Bin) 크기

이클에 균일하게 분포해야 한다. 2번 디스크의 경우 두 개의 마이너 사이클로 빈(bin)이 구성되어 있다. 따라서 2번 디스크에 속한 데이타들은 반으로 나뉘어져서 마이너 사이클에 다시 할당된다. 데이타 c를 먼저 마이너 사이클에 할당한 후 b를 다음 마이너 사이클에 할당함으로써 선형순서를 유지하면서 각 마이너 사이클에 데이타를 균일하게 분포시킨다. 그림 18의 디스크 3의 경우 데이타 *d e i j h f g*는 Modified TS에 의한 선형순서 *a d e i c j b h f g*를 따르고 있고 이 순서대로 차례로 빈에 할당 된다.

빈에 할당된 데이타는 선형순서를 참조하여 하나의 마이너 사이클을 구성한다. *Modified TS*에 의한 선형순서 *a d e i c j b h f g*를 참조로 하여 마이너 사이클 안에서의 데이타 위치를 정한다. 즉, *Modified TS* 혹은 BFS나 DFS를 이용하여 얻은 선형순서를 참조하여 선형순서 상에서 순서가 빠른 데이타를 순서가 나중인 데이타보다 먼저 위치시킨다. 마지막으로 마이너 사이클 안에서 데이타의 위치가 정해진 후에는 시맨틱 관계를 세밀하게 반영하기 위해 최종적으로 마이너 사이클 안에서 데이타 위치 조정을 한다. 그림 18과 같이 조합된 마이너 사이클은 선형 순서를 참조하여 만들어진 것이지만 실제 데이타들의 직접적인 시맨틱 관계나 가중치가 높은 관계를 가지는 데이타간의 직접적인 간선(direct edge)은 고려되지 않은 문제점이 있다. 두 번째 마이너 사이클 *(a i j b)*에서 *a*와 *b*는 직접적인 시맨틱 관계($a \rightarrow b$)가 있지만, *j*와 *b*는 직접적인 시맨틱 관계가 없다. 따라서 시맨틱 관계를 최대한 반영하기 위해 두 번째 마이너 사이클을 *(a, b, i, j)*의 순서로 조정한다. 그림 19는 선형 순서에 의해 빈에 데이타를 할당한 후에 시맨틱 그래프 상 데이타 사이의 간선(direct edge) 유무에 따른 전송 순서 조정을 보인 것이다.

직접 간선을 고려하여 마이너 사이클을 조정하는 방법은 다음과 같다. 하나의 데이타에 대해 직접적인 시맨틱 관계가 있는 데이타들을 하나의 집합으로 하고, 이 집합에 속한 데이타들 중에서 일부가 현재 마이너 사이클 안에 있다면 그러한 데이타는 가까운 거리에 있도록 마이너 사이클 안에서의 데이타들의 순서를 바꾸어 준다. 그리고 가중치가 다양한 값을 가지는 경우에는 가중치가 높은 데이타를 보다 가까운 거리에 있도록 한다. 우리가 제안하는 하이브리드 브로드캐스트 스케줄 작성 단계와 알고리즘은 그림 20과 그림 21에서 제시하였다. 그림 21의 알고리즘은 N이 데이타의 수라고 할 때, 접근빈도에 따라 빈을 할당하는 부분(O(N))과 시맨틱 그래프에서 위상정렬 하는 부분($O(N^2)$), 마지막으로 직접 간선을 반영하기 위해 마이너 사이클을 조정하는 부분($O(N^2)$)으로 나뉘는데, 이러한 작업들이 순차적으로 발생하므로 전체 복잡도는 $O(N^2)$으로 기존 알고리즘에 비해 크게 증가하지 않는다.

완성된 브로드캐스트 스케줄의 최종 비용은 표 4에서 계산하였다. *Modified TS*를 이용하여 시맨틱 관계만을 고려한 스케줄의 비용과 선형순서와 접근 빈도를 함께 고려한 최종 브로드캐스트 스케줄의 총 비용을 비교하

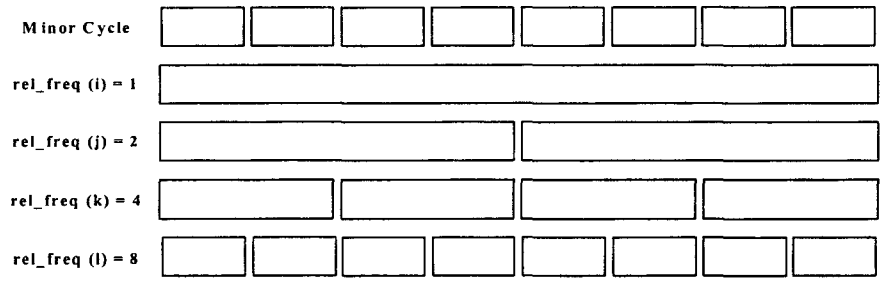



그림 17 데이타 반복 회수에 따른 빈(Bin) 할당

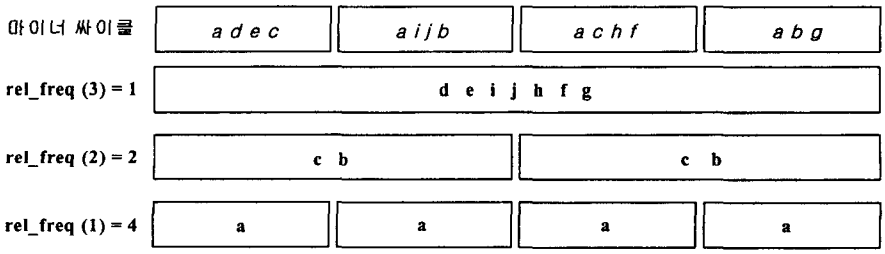



그림 18 Modified TS에 의한 선형순서를 참조한 데이타 할당

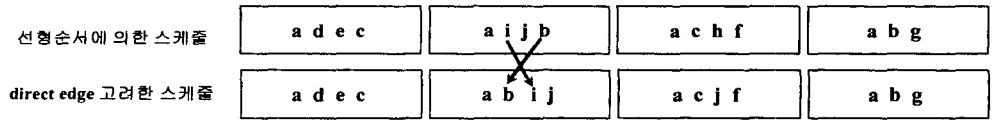



그림 19 마이너 사이클 조정

표 4 완성된 스케줄의 증가 비용

| 노드 선택 방법 | 선형순서 | 개별 비용 중 최소값 | | | | | | | | | | | 총 비용 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | ab | ac | ad | ae | bf | bg | bh | ch | cj | ei | ij | |
| *Modified TS* | *adeicjbhfg* | 6 | 4 | 1 | 2 | 2 | 3 | 1 | 3 | 1 | 1 | 2 | 26 |
| 최종 브로드캐스트 스케줄 | *adecabijachfabg* | 1 | 1 | 1 | 2 | 1 | 1 | 3 | 1 | 3 | 3 | 1 | 18 |

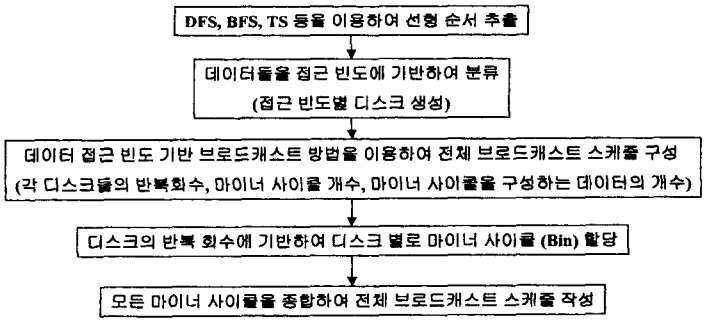



그림 20 하이브리드 브로드캐스트 스케줄 작성 순서

```
input { data set, graph }
 1  DFS, BFS, Modified TS 등을 이용하여 그래프에서 선형순서 생성한다.
    단 그래프 상에서 간선의 가중치 값이 높은 데이타에 우선 순위를 두어
    먼저 방문하도록 한다.
 2  데이타들을 접근 빈도의 내림차순으로 정렬한다.
 3  비슷한 데이타 접근 빈도를 가지는 데이타들을 디스크(disk)로 묶어서 여러 개의 디스크를 생성
    한다.
 4  각 디스크들의 상대적인 접근 빈도를 결정한다.
 5  각 디스크들의 bin 크기를 결정한다. (단위 : 마이너사이클)
      the number of minor cycle m = LCM of the relative frequencies
      the disk's bin size = the relative frequencies of disk / m
 6  선형순서를 참조하여 각 디스크들의 bin에 각 디스크에 속한 데이타들을 할당한다.
 7  각 bin을 구성하는 마이너사이클에 데이타들을 균등하게 할당한다.
    단 참조한 선형순서를 지키면서 할당한다.
10  동일한 위치에 있는 마이너사이클들을 종합하여 전체 브로드캐스트 스케줄을 구성한다.
    마이너사이클 종합 방법
    i. 1)에서 생성한 선형순서를 참조하여 각 마이너 사이클 안에서의 데이타 순서를 결정한다.
    ii. 그래프 상에서의 직접적인 시맨틱 관계를 올바르게 반영하기 위해 다음과 같이한다.
     set the order index in a minor cycle
     for i := 1 to the number of minor cycle
        for j := 1 to the number of minor cycle's data
           the related data set with data d {} = search the graph from data d
           if ( data in the related data set with data d = data in the minor cycle )
              arrange the data by weight in a minor cycle
           end if
        end for
     end for
```

그림 21 하이브리드 브로드캐스트 방법의 스케줄 작성 알고리즘

였다. 우리가 제시하는 하이브리드 브로드캐스트 스케줄링 알고리즘이 만들어 내는 최종 브로드캐스트 스케줄의 시맨틱 비용은 *Modified TS*에 비해 오히려 감소하였다. 이것은 개별 비용만을 비교하였고, 여러 개의 개별 비용 중 최소 값을 선택하였기 때문이다.

## 4. 성능평가 및 결과

이 장에서는 기존의 브로드캐스트 방법과 이 논문에서 제안한 하이브리드 브로드캐스트 방법의 성능을 실험을 통해 비교 평가한다. 모의실험을 통해 다음의 다섯 가지 알고리즘을 비교하였다.

i) Hybrid Broadcast TS - Modified TS로 추출한 선형순서를 사용하는 하이브리드 브로드캐스트 기법, 간선의 가중치를 고려한다.

ii) Hybrid Broadcast DFS - 간선의 가중치를 고려하는 DFS로 추출한 선형순서 사용
DFS 방식으로 그래프를 탐색할 경우 방문할 노드가 다수이면 가중치가 높은 노드를 우선 방문한다.

iii) Hybrid Broadcast BFS - 간선의 가중치를 고려하는 BFS로 추출한 선형순서 사용
BFS 방식으로 그래프를 탐색할 경우 방문할 노드가 다수이면 가중치가 높은 노드를 우선 방문한다.

iv) Broadcast Disk - 데이타의 접근 빈도에 기반한 Acharya의 브로드캐스트 기법

v) Semantic Broadcast - 데이타의 시맨틱 관계에 기반한 Hurson의 브로드캐스트 기법

위의 알고리즘을 클라이언트의 접근 시간(access time) 측면에서 비교함으로써 우리가 제안한 방법이 효율적임을 보인다. (클라이언트의 접근 시간은 클라이언트가 현재 데이타가 브로드캐스트 되고 있는 채널을 듣기 시작해서 원하는 데이타들을 모두 얻을 때까지 걸리는 시간을 의미한다.) 실험에서 사용하는 모든 시간의 단위는 브로드캐스트 단위(broadcast unit)이며, 이것은 서버에서 한 개의 데이타를 전송하는데 걸리는 논리적 시간(logical time)이다. 성능 비교를 위해 각 방법을 CSIM 18 simulator사용하여 구현하였다. CSIM 18 simulator는 시뮬레이션에 널리 쓰이는 프로세스 기반의 시뮬레이션 엔진으로 실험을 쉽게 할 수 있도록 여러 데이타 타입과 함수를 지원한다[16,17]. 표 5와 같은 환경에서 실험을 수행하였다.

### 4.1 클라이언트 실행 모델

성능평가에 사용하는 클라이언트의 변수는 표 6에 정리하였다.

*Think_time*은 클라이언트가 필요한 데이타를 요청하는 간격을 나타낸다. 이 값이 작을수록 서버에게 데이타를 자주 요청한다. 클라이언트는 데이타를 0번부터 *Access_Range*-1까지의 범위 안에서 참조한다. 이것은 브로드캐스트 되는 데이타의 부분 집합이다. 이 범위의 밖에 있는 데이타들은 클라이언트가 참조할 확률이 0인 데이타들이고, 범위 안에서 클라이언트의 데이타 접근 확률은 Zipf distribution을 따른다. Zipf distribution에 따라 데이타 0번이 가장 자주 참조되는 데이타이고, *Access_Range*-1번의 데이타는 거의 참조되지 않는 데이타이다. Zipf distribution은 클라이언트의 균일하지 않은(non-uniform) 데이타 접근 경향을 모델링 하기 위해 많이 사용되는 변수로서 브로드캐스트에 사용될 데이타를 접근 빈도에 따라 정렬했을 때 θ값에 따라 상대 접근 빈도(*rel_freq : relative* access *frequency*)가 높은 데이타를 접근할 확률이 상대 접근 빈도가 낮은 데이타를 접근할 확률보다 높은 분포를 보인다[11-13]. 다시 말하면, Zipf distribution에서는 θ의 값이 증가함에 따라 치우친(skewed) 접근 분포를 나타낸다. θ값은 모든 실험에서 0.5(0 ~ 1)로 하였으며, 이 경우 많은 클라이언트들이 접근 빈도가 높은 데이타들을 참조하려 한다.

클라이언트들의 데이타 접근 형태는 두 가지 종류로 나눌 수 있다. 클라이언트는 시맨틱 관계에 따른 데이타 접근 형태를 나타내거나, 데이타의 시맨틱 관계와는 상

표 5 실험 환경

| Machine | OS | Processor | Memory |
|---|---|---|---|
| Sun Blade 1000 | Solaris 8 | 750 MHz UltraSPARC-III | 1G |

표 6 클라이언트 변수 설정

| Parameter | Description |
|---|---|
| *Num_Client* | 데이타를 필요로 하는 전체 클라이언트의 수 |
| *Think_time* | 클라이언트가 데이타를 참조하는 시간 간격<br>(단위 : broadcast unit) |
| *Access_Range* | 전체 데이타들 중에서 클라이언트가 참조하는 영역 |
| *SAccess_Pattern* | 전체 클라이언트들 중에서 시맨틱 접근 형태로 데이타를 참조하는 클라이언트가 차지하는 비율 |
| *θ* | Zipf distribution parameter |

관없이 무작위(random)로 데이타를 접근한다. 후자의 경우는 클라이언트가 시맨틱 관계가 없는 작은 수의 데이타만을 필요로 하는 경우이다. *SAccess_Pattern*의 값은 시맨틱 관계에 따른 데이타 접근 형태를 가지는 클라이언트들이 전체 클라이언트들에서 차지하는 비율을 나타낸다. *SAccess_Pattern*의 값이 클수록 시맨틱 관계에 따른 접근 형태를 가지는 클라이언트들이 많다는 것을 의미한다. 모든 실험에서 클라이언트의 개수는 1000개로 하여 실험하였다.

### 4.2 서버 실행 모델

서버 실행 모델의 변수는 표 7에 정리하였다. 서버가 브로드캐스트 하는 데이타의 수는 *ServerDBSize*이다. 이 값은 클라이언트가 참조하는 데이타의 수 - 클라이언트 실험 모델 변수인 *Access_Range* 보다 크다. *NumDisks*는 데이타 접근 빈도에 따라 분류한 디스크의 개수이며, *DiskSize_i*는 디스크 *i*에 속해 있는 데이타 개체들의 개수이다. *Num_Related_DATA*는 그래프를 구성하는 데이타들의 개수를 나타낸다. 최대 값은 서버에 있는 데이타의 개수, *ServerDBSize*이며, 이 값이 클수록 많은 데이타들이 시맨틱 관계를 가지고 있음을 의미한다. *Num_Graph*는 사용한 그래프의 개수를 의미한다. 특정한 한 그래프에 대해서만 실험한 것이 아니라 여러 개의 그래프를 사용하여 실험하여 좀 더 일반적인 성능을 알아보고자 하였다. *Data_Frequency_Ratio*는 디스크들의 상대적인 빈도수를 나타낸다. 이 값에 따라 전체 브로드캐스트 스케줄의 크기와 마이너 사이클의 개수도 정해진다. *Noise*는 서버로부터 전송되는 데이타와 클라이언트가 필요로 하는 데이타들의 차이를 비율로 나타낸 것이다. *Noise*가 0의 값을 가지면 서버로부터 브로드캐스트 되는 데이타와 클라이언트가 원하는 데이타가 일치함을 의미한다.

표 7 서버 변수 설정

| Parameter | Description |
|---|---|
| *ServerDBSize* | 브로드캐스트 되는 전체 데이타의 개수 |
| *NumDisks* | 디스크의 개수 |
| *Num_Related_DATA* | 시맨틱 관계를 가지는 데이타들의 개수 |
| *Num_Graph* | 사용한 그래프의 개수 |
| *Disk_Frequency_Ratio* | rel_freq(i) : rel_freq(j), rel_freq(k) ... |
| *DiskSize_i* | disk i의 크기(단위 : 데이타 개수) |
| *Noise* | % workload deviation |

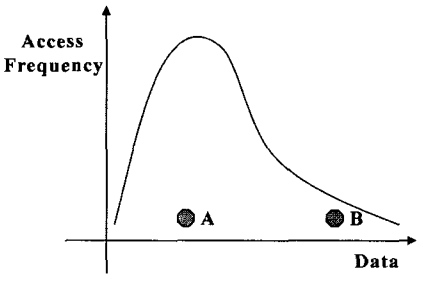



그림 22 데이타의 접근 빈도 수 분포

클라이언트들이 필요로 하는 데이타들의 접근 빈도수의 분포는 그림 26과 같이 특정 데이타들만을 자주 접근하는 경향을 나타낸다. 시맨틱 관계를 나타내는 그래프는 이러한 경향을 반영하여 구성하였다. 예를 들면, 그림 22에서 보이는 데이타 A는 멀리 떨어져 있는 데이타 B와 시맨틱 관계가 있는 것이 아니라, 데이타 A의 주변에 있는 데이타들과 시맨틱 관계가 있다고 설정하였다. 그리고 한 노드로부터 나갈 수 있는 간선의 개수는 최대 6개로 제한하여 그래프를 구성하였다. 모든 실험에서 서버가 브로드캐스트 하는 데이타의 개수는 6000개이며, 이 값은 데이타 개수의 최대 값이다. 그리고 Noise는 0으로 설정하여 실험하였다. 5개의 그래프를 대상으로 하여 실험하였으며, 한 그래프에 대해 10번 실행하였다. 그리고 실험결과, 즉 클라이언트 접근시간은 모두 합쳐서 평균값으로 계산하여 나타내었다. 그리고 모든 그래프에서 간선의 가중치는 다양한 값을 가지도록 하였다.

### 4.3 클라이언트 접근 형태에 따른 접근시간 측정

클라이언트들이 나타내는 접근 형태에 따라 각 클라이언트의 접근시간은 다르게 된다. 클라이언트들은 데이타를 시맨틱 관계를 기반으로 하여 데이타를 참조할 수도 있고 무작위(random)로 데이타를 참조하려 할 수도 있다. 이 실험에서는 무작위의 데이타 접근 형태를 가지는 클라이언트들이 전체 클라이언트에서 차지하는 비율을 100%, 90%, 80%, ···, 0%로 줄어들게 하며, 클라이언트들의 접근시간을 측정하였으며, 클라이언트들의 데이타 접근 형태는 Zipf distribution에 따른다.

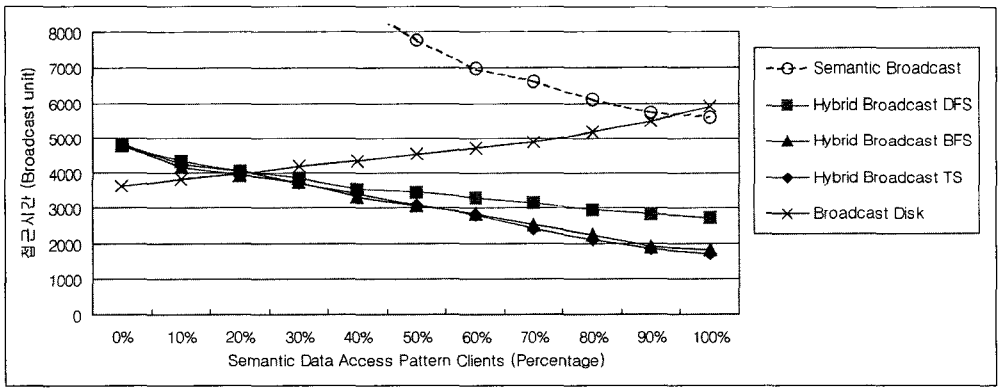
그림 23은 접근 형태 비율을 달리하여 각각의 브로드캐스트 알고리즘의 성능을 비교한 것이다. 그림 23에서 볼 수 있듯이 하이브리드 브로드캐스트 방법은 클라이

표 8 실험 변수 설정

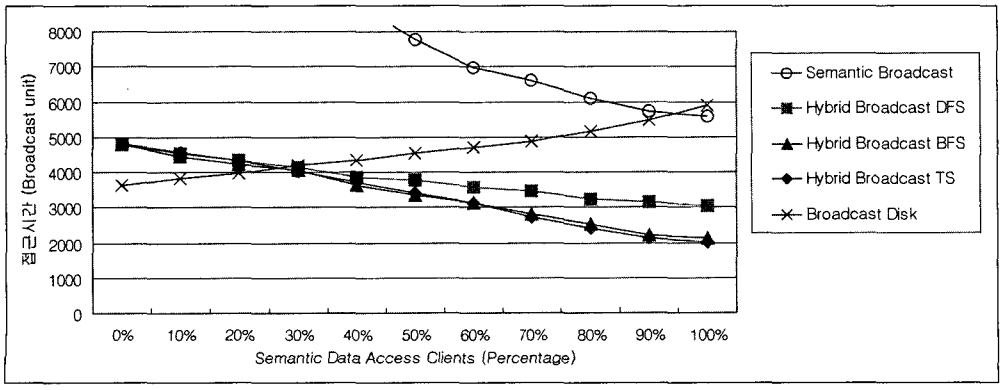
| | | |
|---|---|---|
| *Think time* | 1.0 | |
| *ServerDBSize* | 6000 | |
| *Access Range* | 1200 | |
| $\theta$ | 0.5 | |
| *Num_Ralated_DATA* | 1.0 ( 0 ~ 1.0 ) | |
| *Disk_Frequency_Ratio* | 6 : 5 : 4 : 3 : 2 : 1 | |
| *Max Edge Weight* | 5 (1, 2, ..., 5) | 10 (1, 2, ..., 10) |
| *Noise* | 0 | |
| *SAccess_pattern* | 0.0, 0.1, 0.2, ... 0.9, .1.0 | |

언트의 데이타 접근 형태에 관계없이 비교적 좋은 데이타 접근시간을 보이고 있다. 시맨틱 브로드캐스트 방법은 시맨틱 관계를 기반으로 하여 데이타를 참조하는 클라이언트가 많아질수록 성능이 좋아지긴 하지만, 데이타들의 빈도 수를 반영하지 않고, 모든 데이타들을 한 번씩만 전송하기 때문에 데이타를 무작위로 접근하는 클라이언트가 많을 때는 가장 최악의 성능 시간을 가진다. 이외에 디스크들의 상대적인 빈도 수를 (4:2:1)로 한 상태에서도 실험해보았지만, 실험결과는 접근시간에서만 차이가 있을 뿐 비슷한 형태를 나타내었다. BFS를 이용한 하이브리드 브로드캐스트 방법과 TS를 이용한 하이브리드 브로드캐스트 방법은 거의 비슷한 성능을 보이면서, 때때로 접근시간이 서로 역전되는 상태를 보여주는데, 이것은 클라이언트의 데이타 접근 형태가 BFS 혹은 TS 중에서 어느 하나에 유리하게 나타나기 때문으로 보인다. 그리고 DFS를 이용한 방법은 BFS와 TS를 이용한 것보다 떨어지는 성능을 보이는데, 이것은 DFS에 의한 선형 순서가 개별 클라이언트의 데이타 접근 형태를 나타내는데는 적합하지만, 전체 클라이언트들의 데이타 접근 형태를 종합하여 나타내기에는 부족함이 있기 때문으로 판단된다. DFS에 의한 선형순서는 클라이언트의 개별적인 데이타 접근 비용을 감소시키는 데 비해, BFS와 TS에 의한 선형순서는 전체적인 클라이언트의 데이타 접근 비용을 감소시키는 차이점이 있기 때문이다.

간선의 가중치는 한 데이타를 접근한 클라이언트가 간선으로 연결된(시맨틱 관계가 있는) 데이타를 접근하는 비율을 나타낸다. 가중치의 최대 값이 5(1, 2, 3, 4, 5)라고 가정하고 한 간선의 가중치가 5라면 그러한 간선으로 연결된 데이타는 항상 클라이언트가 참조한다는 것을 의미한다. 즉, 연결된 간선의 가중치 값이 큰 데이타는 클라이언트가 간선으로 연결된 다른 데이타들보다 자주 접근한다는 것을 의미한다. 그림 23의 실험에서는 가중치를 고려하여 선형순서를 생성하였으며, 이러한 가중치가 반영되었기 때문에 브로드캐스트 디스크 방법보다 하이브리드 브로드캐스트 방법이 좋은 성능을 보이




(a) 가중치(Weight)의 최대값 : 5




(b) 가중치(Weight)의 최대값 : 10

그림 23 접근 형태 비율에 따른 클라이언트 접근시간

고 있다. 그리고 가중치의 최대값이 5인 경우가 10인 경우보다 약간 더 좋은 성능을 보이고 있는데 이것은 가중치의 구분 단계가 작기 때문인 것으로 판단된다.

### 4.4 데이타의 시맨틱 관계 비율에 따른 접근시간 측정

시맨틱 관계를 가지는 데이타가 전체 데이타들 중에서 차지하는 비율에 따라 클라이언트의 접근시간이 영향을 받는다. 그림 24는 시맨틱 관계를 가지지 않는 데이타가 브로드캐스트 되는 전체 데이타에서 차지하는 비율을 0%, 10%, 20%, … 100%로 변화시키며 클라이언트의 접근 시간을 측정한 것이다.

표 9 실험 변수 설정

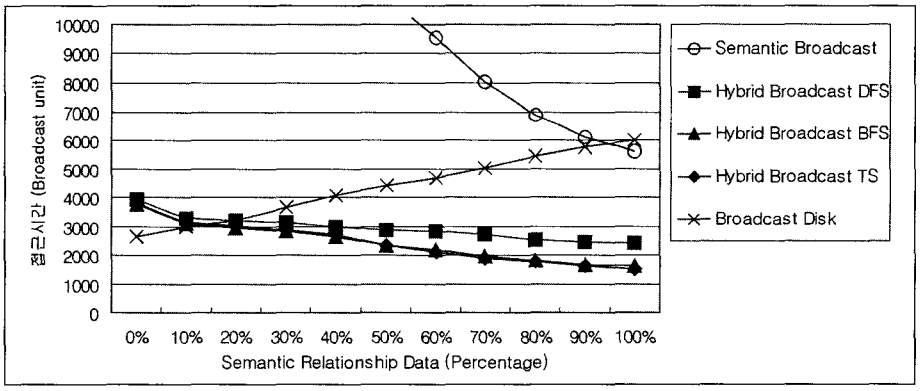
| | | |
|---|---|---|
| *Think time* | 1.0 | |
| *ServerDBSize* | 6000 | |
| *Access Range* | 1200 | |
| $\theta$ | 0.5 | |
| *Num_Ralated_DATA* | 0.0, 0.1, 0.2, ... , 1.0 | |
| *Disk_Frequency_Ratio* | 6 : 5 : 4 : 3 : 2 : 1 | |
| *Max Edge Weight* | 5 (1, 2, ..., 5) | 10 (1, 2, ..., 10) |
| *Noise* | 0 | |
| *SAccess_pattern* | 0.5 | |

이 실험에서도 4.3절과 마찬가지로 디스크의 상대적인 빈도 수를 (4 : 2 : 1)로 하여 실험해보았지만 마찬가지로 접근시간의 차이만 있었을 뿐 그림 24와 같은 접근 시간을 나타내었다.

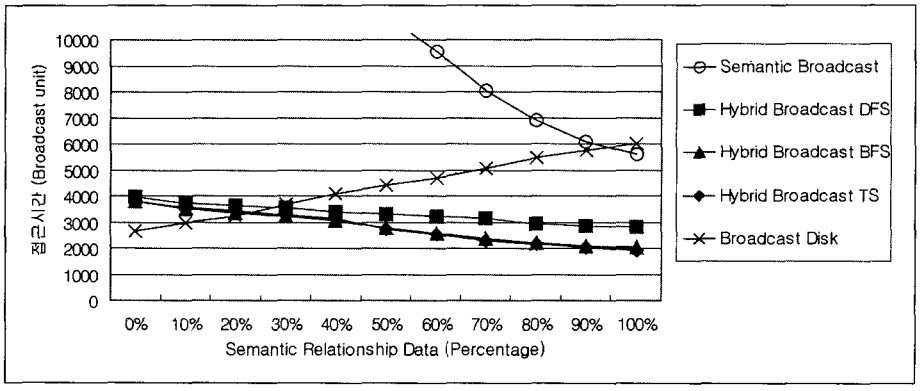
하이브리드 브로드캐스트 방법은 브로드캐스트 디스크 방법과 마찬가지로 데이타들의 빈도 수를 반영하고 있지만, 시맨틱 관계도 고려하였기 때문에 그림 24에서 보듯이 시맨틱 관계를 가진 데이타가 많아질수록 브로드캐스트 디스크보다 좋은 성능을 보이고 있다. 시맨틱 관계를 가지는 데이타들의 개수가 적을 때에는 기존 브로드캐스트 디스크 방법이 가장 좋은 성능을 보이는데, 이것은 브로드캐스트 디스크 방법은 전송되는 데이타들의 반복 주기가 일정하기 때문이다.

### 4.5 클라이언트 데이타 접근 형태와 브로드캐스트 스케줄 차이에 따른 영향

클라이언트의 데이타 접근 형태와 브로드캐스트의 선




(a) 가중치(Weight)의 최대값 : 5




(b) 가중치(Weight)의 최대값 : 10

그림 24 시맨틱 관계의 데이타 비율에 따른 클라이언트 접근시간

형 스케줄이 어떤 형태를 나타내느냐에 따라 접근시간이 변화될 수 있다. 현재 데이타가 브로드캐스트 되고 있는 스케줄은 BFS를 이용하여 생성된 선형순서를 참조하였다고 가정하자. 하지만 만약 클라이언트의 데이타 접근 형태가 그래프를 DFS 방식으로 탐색하는 형태를 나타낸다면, 클라이언트의 데이타 접근 형태와 데이타의 브로드캐스트 방식이 다르기 때문에 클라이언트의 접근시간이 길어지게 된다. 다음 실험은 이러한 영향, 즉 클라이언트의 데이타 접근 형태와 선형순서의 차이가 클라이언트 접근시간에 미치는 영향을 알아보기 위해 실행한 것으로써, 실험 변수는 표 10과 같이 설정하였다.

클라이언트의 데이타 접근 형태를 그림 25와 같이 DFS, BFS, TS의 세 가지로 구분하였다. 그리고 각 접근 형태에 따라 클라이언트가 데이타를 요구할 때 그래프 상에서 depth를 깊게 해서 들어가는 접근 형태(*Deep*)와 얕은 depth의 관련 데이타에서 탐색을 끝내는 형태(*Shallow*)로 분류하였다. 따라서 클라이언트의 데이타 접근 형태는 총 6가지로 분류된다. Depth의 최대값은 13으로 하였으며, 클라이언트의 데이타 접근 깊이가 (Depth ≤ 4)이면 Shallow, (Depth > 4)이면 Deep으로 구분하였다 그림 26은 클라이언트의 데이타 접근 형태를 6가지로 분류하였을 때 각 브로드캐스트 방법에 대해 클라이언트의 접근시간을 측정한 것이다.

실험 결과, BFS와 TS를 이용한 하이브리드 브로드캐스트는 비슷한 성능을 보여주었지만, 클라이언트가 DFS 기반의 데이타 접근 형태를 가질 때는 접근시간이 길어지는 현상을 보였으며, 클라이언트의 데이타 접근 형태와 데이타의 브로드캐스트 상태가 일치할수록 성능이 좋게 나타났다. 그리고 시맨틱 브로드캐스트 방법은 비록 DFS를 기본으로 선형순서를 생성하기는 하지만, 데이타 접근 빈도가 반영되지 않았기 때문에 다른 방법들보다 긴 접근시간을 나타내었다.

표 10 실험 변수 설정

| | |
|---|---|
| *Think time* | 1.0 |
| *ServerDBSize* | 6000 |
| *Access Range* | 1200 |
| $\theta$ | 0.5 |
| *Num_Ralated_DATA* | 1.0 ( 0 ~ 1.0 ) |
| *Disk_Frequency_Ratio* | 6 : 5 : 4 : 3 : 2 : 1 |
| *Noise* | 0 |
| *SAccess_pattern* | 1.0 |
| *Max Depth* | 13 |

## 5. 결 론

이동 컴퓨팅 환경에서 효율적인 데이타 전송을 위해 사용되는 브로드캐스트 방법에는 여러 가지가 있지만, 기존의 방법은 데이타의 접근 빈도만을 고려하거나, 데이타들의 시맨틱 관계만을 고려한 것이었다. 이 논문에서는 데이타들의 시맨틱 관계와 접근 빈도를 함께 반영하여 데이타 브로드캐스트 스케줄을 작성하는 하이브리드 브로드캐스트 방법을 제안하였다. 이 방법은 각 데이타들의 접근 빈도에 따른 데이타의 반복 전송을 위해 기존의 브로드캐스트 디스크 방법을 기본적으로 사용하여 먼저 전송 회수가 다른 각 디스크들을 구성한다. 또한, 그래프로 표현된 시맨틱 관계를 반영하기 위해 그래프를 선형순서로 변환하여 이 선형순서에 따라 각 디스크에 속해 있는 데이타들 전송 순서를 결정하여 전체 브로드캐스트 스케줄을 작성한다.

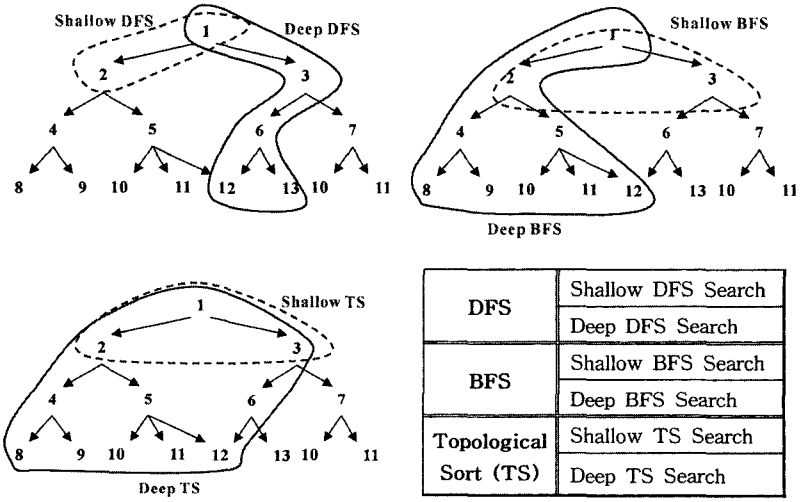



그림 25 클라이언트 데이타 접근 형태 분류

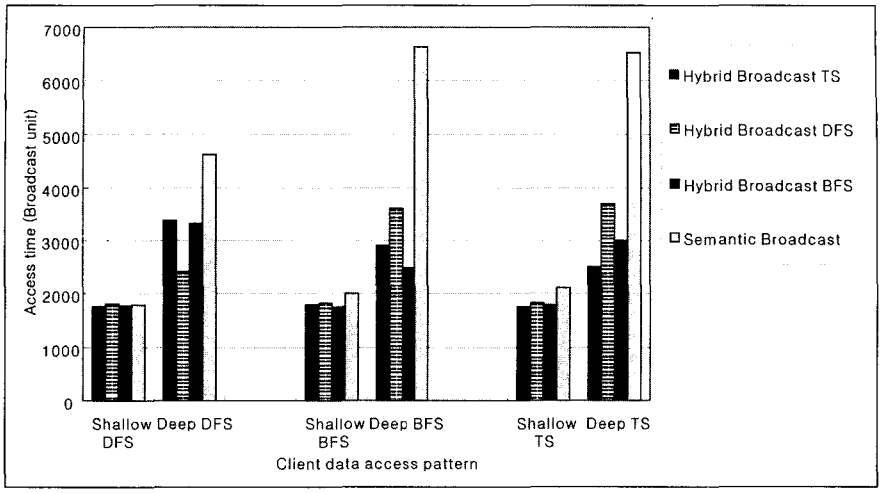



그림 26 클라이언트 데이타 접근 형태에 따른 접근시간

이 논문에서 제안된 하이브리드 브로드캐스트 방법이 클라이언트의 데이타 접근시간을 감소시킨다는 것을 데이타의 유형과 데이타 접근 형태를 달리하는 여러 가지 실험을 통해 보여주었다. 우리가 제안한 하이브리드 브로드캐스트 방법은 데이타의 시맨틱 관계나 클라이언트의 데이타 접근형태에 크게 좌우되지 않고 좋은 성능을 보여주었다. DFS는 BFS, TS에 비해 떨어지는 성능을 보여주었는데 이것은 DFS를 이용한 방법이 개별 클라이언트의 데이타 접근 형태는 만족시키지만, 전체 클라이언트들의 데이타 접근 형태를 종합하여 만족시키기에는 부족함이 있기 때문으로 판단된다. 따라서 BFS 혹은 TS를 사용한 하이브리드 브로드캐스트 방법을 사용하며, 둘 중에 어느 방법을 사용할 지는 클라이언트들의 데이타 접근 형태가 어떤 형태를 나타내느냐에 따라 결정한다.

향후 이 연구는 다음과 같이 확장될 수 있다. 첫째, 이 논문은 서버에서 데이타를 브로드캐스트 하는 채널을 하나로 설정하였지만, 전송 채널이 여러 개가 있을 경우 효율적인 브로드캐스트 기법에 대한 연구, 즉 데이타 브로드캐스트 채널을 여러 개 사용할 경우, 여러 개의 채널에 데이타들의 시맨틱 관계를 반영하면서 각 채널에 데이타를 분배하는 방법에 대한 연구가 필요하다. 둘째, DFS, BFS, TS 외에 다른 선형순서 생성 방법에 대한 추가적인 연구가 필요하다. 이 논문에서, 시맨틱 비용은 두 데이타 사이의 간선에 대한 총 비용만을 고려하였다. 클라이언트의 데이타 베이스 질의가 대부분 그래프 경로상의 데이타를 순차적으로 필요로 하는 경우 이를 효율적으로 만족시키기 위해서 그래프상의 전체 경로를 고려한 선형 순서 생성이 필요할 것이다. 셋째, 이 논문은 브로드캐스트 스케줄을 생성할 때 인덱스는 고려하지 않았다. 인덱스를 함께 브로드캐스트 하는 경우, 밀접한 시맨틱 관계를 맺고 있으면서 빈번하게 접근되는 클러스터에 대한 인덱스를 효율적으로 사용하여 접근시간과 튜닝시간을 줄일 수 있다. 마지막으로 본 연구의 실험은 접근 시간에 대한 성능만을 측정하였다. 인덱스를 사용하는 경우, 접근 시간과 함께 튜닝 시간에 대한 성능 비교가 필요하다.

## 참 고 문 헌

[1] G. Forman and J. Zahorjan, "The Challenges of Mobile Computing," IEEE Computer, Vol. 27, No.6, pp. 38-47, April 1994.

[2] T. Imielinski and B. Badrinath, "Wireless Mobile Computing : Challenges in Data Management," Comm. 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] 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.

[5] J. Banerjee, W. Kim, S. Kim, J. Garzz, "Clustering a DAG for CAD Databases," IEEE transaction on Software Engineering, Vol.14, No.11, November 1988.

[6] K. C. K. Lee, H. Leong, A. 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.
[7] J. Juran, A. Hurson, N. Vijaykrishman and S. Boonsiriwattanakul, "Data Organization and Retrieval on Parallel Air Channels, Performance and Energy Issues," Proc. of International Conference on High Performance Computing(HiPC 2000), pp. 501-510, 2000.
[8] A. Hurson., Y. Chehadeh, J. Hannan, "Object organization on parallel broadcast channels in a global information sharing environment," Proc. of IEEE International Performance, Computing, and Communications Conference(IPCCC'00), pp. 347-353, 2000.
[9] K. C. K. Lee, H. Leong, A. Si, "Semantic Data Access in an Asymmetric Mobile Environment," Proc. of the 3rd International Conference on Mobile Data Management(MDM'02), 2002.
[10] A. Hurson, Y. Chehadeh, L. Miller, "Object organization on a single broadcast channel in a global information sharing environment," Proc. 24th Euromicro Conference, pp. 1021-1028, 1998.
[11] D. Knuth, The Art of Computer Programming, *Second Edition*, Vol III, Addison Wesley, 1998.
[12] G.K. Zipf. Human Behaviour and the Principle of Least Effort : An Introduction to Human Ecology. Addison Wesley Press, Cambridge, Massachusetts, 1949.
[13] J. Gray, P. Sundaresan, S. Englert, K. Baclawski, P. Weinberger, "Quickly generating billion-record synthetic databases," Proc. of the ACM SIGMOD International conference on Management of data, 1994.
[14] Y. Chung, M. Kim, "An index replication scheme for wireless data broadcasting," The Journal of Systems and Software, Vol.51, pp. 191-199, 2000.
[15] Y. Chung, M. Kim, "Effective data placement for wireless broadcast," Distributed and Parallel Databases, Vol.9, pp. 133-150, 2001.
[16] H. D Schwetman, "CSIM : A C-based process oriented simulation language," Proc. Winter Simulation Conference, 1986.
[17] CSIM18 Simulation Engine USER'S GUIDE, Mesquite Software, Inc.
[18] 이송이, 정성원, "단일 무선 채널에서 객체 간 선형 순서를 보장하는 효율적인 브로드캐스트 스케줄링 기법", 제13회 통신 정보 합동 학술대회, May 2003.

최 성 환
2003년 서강대학교 컴퓨터학과 대학원 졸업. 2001년 서강대학교 컴퓨터학과 졸업. 관심분야는 이동 컴퓨팅 시스템

정 성 원
정보과학회논문지 : 데이타베이스
제 30 권 제 3 호 참조

이 송 이
1988년 이화여자대학교 전자계산학과, 이학사. 1990년 미시건 주립대학교(Michigan State University) 전산 과학과(Department of Computer Science) 공학 석사. 1997년 서울대학교 컴퓨터 공학과, 공학박사. 1997년~1998년 미국 University of Wisconsin-Madison, Post-Doc. 1999년~2000년 서울대학교 중앙교육연구전산소 특별연구원. 2000년~현재 성신여자대학교 컴퓨터 정보학부 계약교원