다중 방송채널을 위한 데이타 할당
(Data Allocation for Multiple Broadcast Channels)

정 성 원 [†] 남 승 훈 [††] 정 호 련 [†††] 이 원 택 [†††]
(Sungwon Jung) (Seunghoon Nam) (Horyun Jeong) (Wontaek Lee)

요 약 무선 환경에서는 채널의 대역폭과 휴대장치의 전력이 제한된다. 이러한 환경에서 데이타를 브로드캐스트(Broadcast) 하는 것은 효과적으로 클라이언트(Client)에게 데이타를 제공하기 위한 훌륭한 기법이 된다. 싱글 채널(Single channel)에서 뿐만 아니라 다중 채널(Multi-channel) 환경에서 클라이언트들의 데이타에 대한 접근 패턴에 따라 브로드캐스트 프로그램을 작성하는 기법이 연구되어 왔다. 본 논문에서는 물리적으로 독립된 다중 채널 환경에서의 효과적인 브로드캐스트 기법에 대해 소개한다. 기존에 소개된 기법들에서는 데이타의 접근 확률(access probability)을 기초로 주어진 물리적 채널 수만큼 데이타 집합을 분할하게 된다. 이러한 기법들은 높은 접근 확률을 가지는 소수의 데이타와 낮은 접근 확률을 가지는 다수의 데이타로 나누게 된다. 하지만 이러한 기법들은 단순히 채널 수에 따른 데이타의 구분에만 관심이 있기 때문에, 동등한 채널 내에서 생기는 데이타들 간의 접근 확률의 차이가 무시된다. 실제 환경에서는 많은 수의 물리 채널을 확보하기가 어렵고, 상당히 많은 수의 데이타를 브로드캐스트 해야 한다. 따라서 동등한 채널 내에서도 접근 확률을 기초로 브로드캐스트 프로그램을 작성할 수 있다면 기존의 기법에서보다 더 좋은 성능을 낼 수 있을 것이다. 본 논문에서는 데이타의 집합을 채널 수에 맞게 분할함과 동시에 채널 내에서 브로드캐스트 되는 횟수를 정함으로써 다중 채널에서 접근확률의 편차가 심한 데이타를 브로드캐스트 해야 하는 상황에서 보다 적합한 방법을 제안한다.

키워드 : 모바일 컴퓨팅, 모바일 데이타베이스, 브로드캐스트, 다중 무선 채널, 적응형 데이타 할당방법

Abstract The bandwidth of channel and the power of the mobile devices are limited on a wireless environment. In this case, data broadcast has become an excellent technique for efficient data dissemination. A significant amount of researches have been done on generating an efficient broadcast program of a set of data items with different access frequencies over multiple wireless broadcast channels as well as single wireless broadcast channel. In this paper, an efficient data allocation method over multiple wireless broadcasting channels is explored. In the traditional approaches, a set of data items are partitioned into a number of channel based on their access probabilities. However, these approaches ignore a variation of access probabilities of data items allocated in each channel. In practice, it is difficult to have many broadcast channels and thus each channel need to broadcast many data items. Therefore, if a set of data items broadcast in each channel have different repetition frequencies based on their access frequencies, it will give much better performance than the traditional approaches. In this paper, we propose an adaptive data allocation technique based on data access probabilities over multiple broadcast channels. Our proposed technique allows the adaptation of repetition frequency of each data item within each channel by taking its access probabilities into account.

Key words : Mobile Computing, Mobile Databases, Data Broadcast, Data Dissemination, Multiple Wireless Channels, Adaptive Data Allocation

· 본 연구는 한국과학재단 목적기초연구(R01-2003-000-10197-0)지원과 서강대학교 산업기술연구소의 지원을 받아 수행되었음
† 종신회원 : 서강대학교 컴퓨터학과 교수
jungsung@sogang.ac.kr
†† 비 회 원 : 삼성전자 TN Business 연구원
ourleo@sogang.ac.kr
††† 비 회 원 : 서강대학교 컴퓨터학과
queenking@mclab.sogang.ac.kr
magicsniper@mclab.sogang.ac.kr
논문접수 : 2005년 2월 18일
심사완료 : 2005년 11월 10일

1. 서 론

무선 이동 통신과 휴대 장치의 비약적인 발달에 힘입어 각계각층의 사람들은 무선 이동 통신을 통해 언제 어디서나 쉽게 원하는 정보를 얻을 수 있게 되었다. 그러나 기존의 유선 네트워크에서는 고려할 필요가 없었던 문제들이 무선 환경에 와서는 대두되었다. 그 첫째가 무선 네트워크의 대역폭과, 휴대장치에게 필요한 전력이 제한된다는 점이다[1]. 둘째로 무선 환경의 비대칭성을 들 수 있다. 비대칭성이란 휴대 장치에 필요한 전력이 데이타를 받을 때 보다 데이타를 보낼 때에 월등히 크다는 특징을 말한다. 이러한 문제들로 인하여 전통적으로 유선망에서 사용하던 요구-응답(request-response) 시스템은 더 이상 무선 환경에서 데이타를 클라이언트(Client)에게 전송하기 위한 알맞은 기법이 아니다[2].

이와 같은 특성을 지닌 무선 환경에서 데이타를 전송하기 위하여 다양한 연구 기법들이 소개되었다. 그 가운데에 가장 무선 환경에 적합해 보이는 기법으로 브로드캐스트(Broadcast) 방법이 연구되어 왔다. 브로드캐스트 기법은 서버가 보유한 모든 데이타를, 확보하고 있는 무선의 채널을 통해 주기적으로 뿌려주는 기법을 말한다. 이와 같은 방법을 사용하게 되면, 클라이언트는 자신이 원하는 데이타가 마치 자신이 가진 물리적 디스크에 저장되어 있는 것처럼 가상의 디스크—서버가 데이타를 브로드캐스트하고 있는 채널—에 접근하여 원하는 데이타를 얻게 된다. 따라서 브로드캐스트 기법을 사용하는 시스템에서는 정보를 전송하는 데에 있어서 필요한 비용이 클라이언트의 수와는 무관하게 된다. 이와 같이 클라이언트의 수에 제한이 없다는 장점으로 인해 무선 데이타 브로드캐스트는 경매나 전자 입찰과 같은 전자 상거래 응용 분야와 주식 거래, 그리고 기상 정보나 교통 정보 방송 분야와 같은 다양한 응용 분야에 활용되고 있다. 이것은 무선 환경이 제한된 대역폭을 가진다는 측면에서 매우 유용한 특징이다. 뿐만 아니라 휴대 장치가 원하는 데이타를 얻기 위해 서버에게 추가적인 요청을 할 필요가 없기 때문에 휴대 장치에게 있어서 제한된 소비 전력을 절약할 수 있게 된다. 그러므로 데이타 브로드캐스트는 모바일 환경에서 데이타를 공급하는 데에 효과적인 방법이 될 수 있다. 이제 남은 문제는 클라이언트가 효과적으로 데이타를 얻기 위해서 브로드캐스트 채널에 데이타 아이템을 할당하는 방법이다.

브로드캐스트 채널에 데이타를 할당한 한 주기를 브로드캐스트 프로그램(broadcast program)이라고 한다. 효과적인 브로드캐스트 프로그램을 작성하기 위해서는 다음과 같은 요소가 고려되어야 한다.

접근시간은 클라이언트가 원하는 데이타를 얻기 위하여 브로드 캐스트 채널에 처음 탐색(initial probe)을 시작한 순간부터 최종적으로 원하는 데이타를 획득할 때까지의 총 시간을 말한다. 결국 클라이언트가 데이타를 획득하기 위해서는 접근 시간(access time) 만큼을 소비해야 한다. 이 때문에 접근 시간은 브로드 캐스트 프로그램(broadcast program)의 성능을 측정하는 데에 사용된다.

브로드캐스트 프로그램을 설계하는 데에 있어서 평균 접근 시간을 최소로 하는 것이 중요한 문제가 된다. 클라이언트가 데이타를 얻기 위해 기다리는 시간을 줄일 뿐만 아니라 제한된 대역폭과 휴대 장치의 소비 전력을 절약할 수 있기 때문이다. 싱글 채널(single channel)에서 평균 접근 시간(average access time)을 줄이는 브로드캐스트 프로그램을 작성하기 위한 많은 연구들이 있었고, 싱글 채널(single channel)에서 뿐만 아니라 다중 채널(multi- channel)에서 이와 같은 효과적은 브로드캐스트 프로그램을 작성하기 위한 연구도 시도되고 있다.

브로드캐스트 시스템에서 한 가지 중요한 문제점은 브로드캐스트 채널을 모든 클라이언트가 공유한다는 점이다. 이 때문에 클라이언트는 원하는 데이타에 접근하기 위해서 원하지 않는 데이타가 지나가기를 기다려야 한다. 이러한 문제는 데이타의 접근 패턴이 한 쪽으로 치우쳐질 때 더 악화된다. 이러한 경우 대부분의 클라이언트가 접근하려고 하는 소수의 접근 확률(access probability)이 높은 데이타를 얻기 위해서 다수의 인기 없는 데이타를 전송해야 하는 부담과 이에 따라 평균 접근 시간이 더욱 커지게 된다. 이와 같은 문제를 해결하기 위해서 데이타의 접근 확률(access probability)을 고려하여 보다 인기 있는 데이타를 보다 자주 브로드캐스트 하는 방안이 제시되어 왔다[3].

데이타의 접근 확률에 따라 브로드캐스트 프로그램을 작성하는 많은 연구들은 싱글 채널(single channel)을 가정한 것들이 대부분이었고, 물리적으로 서로 독립된 다중 채널 환경을 가정한 연구[4-7]들도 동일 채널 내에서는 데이타의 접근 확률의 차이가 무시되어 왔다. 기존연구들은 기존 다중 채널 환경을 가정한 연구들이 전체 데이타의 집합을 채널 수의 관점에서 파티션(partition) 하는 문제로 간주하여, 다중 채널 환경일지라도 특정한 한 채널 내에서만 보면 그 채널에 할당된 모든 데이타 아이템들은 브로드 캐스트 주기 안에서 단지 한 번씩만 할당되어 있기 때문에 다중채널임에도 불구하고 클라이언트가 특정한 한 채널을 들을 때는 데이타의 선호도에 상관없이 동일한 채널에 있는 모든 데이타의 응답시간이 동일하였다. 실제 모바일 환경에서는 물리적으로 독립된 다중 채널을 가진다 할지라도 그 수가 매우

작으며, 브로드 캐스트 해야 할 전체 데이타 아이템의 수는 이에 비하여 월등히 크므로 특정한 하나의 채널 내에 할당된 데이타들 간에 접근 확률의 차이가 커질 수 있다. 그러므로 전체 데이타의 집합을 단순히 여러 개의 채널로 파티션 하는 문제에 더하여 한 채널 내에서 다시 접근 확률(access probability)에 따라 더 인기 있는 데이타를 더 자주 할당할 수 있다면, 클라이언트측에서 느끼는 응답시간은 크게 줄어들 것이다.

이 논문에서는 물리적으로 서로 독립된 다중 채널 환경에서 데이타의 접근 확률(access probability)에 따라 브로드캐스트 프로그램을 작성하는 방안을 제시한다. 여기서 제안하는 방법은 데이타의 접근 확률에 근거하여 전체 데이타의 집합을 채널수에 맞게 나누는 것에 더하여, 각 부분 집합 내에서 데이타 아이템들에게 필요한 최적의 고유 할당 수를 찾아내어 동일 채널 내에서도 데이타의 접근 확률이 반영될 수 있도록 하였다. 물론 여기서 제안하는 방법은 동일 채널 내에서 최적의 할당 수를 찾기 위한 추가적인 연산이 필요하기 때문에, 기존에 제시된 방법들에 비하여 보다 많은 수행 시간이 필요하다. 하지만 결과적으로 브로드캐스트 시스템의 평가 요소인 접근 시간을 상당히 낮추는 기여를 하고 있다.

다음 2장에서는 다중 채널 환경에서의 기존 연구들의 특성과 장·단점을 살펴볼 것이다. 살펴볼 연구들은 FLAT[3], Bin-Packing[5], VFk[6] 그리고 Greedy 방법을 사용한 연구[4]가 될 것이다. 이 연구들은 모두 동일한 상황과 가정을 사용한다. 따라서 3장에서는 브로드캐스트 시스템 모델의 보다 형상화된 구체적인 설명과 더불어 가정한 상황들을 대수적으로 정의할 것이다. 또한 새로 정의한 비용 함수(cost function)를 최소화 시키는 방안과 그에 맞는 해를 찾아내는 과정을 보일 것이다. 3장에서 마지막으로 해를 찾아내기 위해 필요한 종료조건을 보이고, 증명할 것이다. 4장에서는 실험 환경과 변수들을 설명하고 수행시간과 평균 접근 시간을 측정한 결과에 대해서 설명할 것이다. 마지막으로 5장에서는 결론과 더불어 향후 연구 과제에 대해서 언급하도록 하겠다.

2. 관련 연구

싱글 채널(single channel)에서 뿐만 아니라 다중 채널(Multi-channel) 환경에서 브로드캐스트 프로그램을 작성하는 연구들이 있었다. 이번 장에서는 이러한 기존의 네 가지—FLAT[3], Bin-Packing[5], VKk[6] 그리고 Greedy 방법을 이용한 연구[4]—에 대해서 간략하게 살펴보고 본 논문에서도 이용한 Greedy알고리즘에 대하여 자세히 설명하도록 하겠다.

다중채널에서 데이타를 할당하는 네 가지 알고리즘들을 정리해 보면 표 1과 같다.

이러한 연구들은 모두 동일한 시스템 모델을 가정하고 있다. 먼저 브로드캐스트 영역 내에 K개의 채널을 가정한다. 각 채널은 C_i, $1 \le i \le K$로 표기한다. 또한 서버가 브로드캐스트 할 데이타의 총 크기를 N으로 가정한다. 즉 서버는 총 N개의 데이타를 K개의 브로드캐스트 채널을 통해 클라이언트에게 공급해야 한다. 각 데이타 아이템은 d_j, $1 \le j \le N$로 표기한다. 각 데이타 아이템은 K개의 채널을 통해 모두 브로드캐스트 되어야 하므로, 채널 i에서 N_i개의 데이타 아이템을 브로드캐스트 한다고 할 때, N_i $(1 \le i \le K)$의 총 합은 전체 데이타 아이템의 크기인 N이 된다. 즉, $\sum_{i=1}^{K} N_i = N$ 이 된다. 각 데이타 아이템 d_j는 자신의 접근 확률을 가지게 되는데, 이것을 p_i로 표기한다. 접근 확률 p_j는 클라이언트들이 데이타 아이템 d_j를 p_i의 확률로 접근함을 나타내는 것이다. 접근 확률 p_j는 각 데이타 아이템 d_j에 대한 이산 확률 값이며, 이것은 특정 확률 분포를 따르게 된다. 따라서 확률 분포가 가지는 특성을 모두 만족하게 된다. 모든 데이타 아이템의 접근 확률 p_j를 합한 값이 1이 된다. 즉 $\sum_{j=1}^{N} p_j = 1$이다. 그림 1은 채널이 4개인 브로드캐스트 시스템의 구조를 보여주고 있다.

지금까지 설명한 브로드캐스트 시스템 모델에 근거하

표 1 다중채널에서 데이타를 할당하는 알고리즘들의 비교

	Flat	Bin-packing	VF^K	Greedy
데이타 할당방법	$\frac{N}{K}$ (N: 데이터의 갯수 K: 채널의 갯수)	$bin = \frac{1}{K}$ 각 채널의 데이타 접근 확률의 합이 bin의 용량이 되도록 데이타 할당	데이타 접근 확률을 바탕으로 할당 트리를 구성	평균 기대 지연값을 최소로 하는 시점을 분할 시점으로 선정
복잡도	$O(K)$	$O(N\log N)$	$O(KN^2\log K)$	$O((N+K)\log K)$
비고	클라이언트의 데이타 접근 확률 분포가 편향될수록 접근 시간이 커진다.	Flat 방법에 비해 데이커 접근 확률이 반영되어있으나, 알고리즘이 단순하다.	최적의 알고리즘인 DP알고리즘과 비슷한 성능을 보이나 시간복잡도가 크다.	DP와 VF^K보다 수행시간이 빠르면서 비슷한 성능을 보여준다.

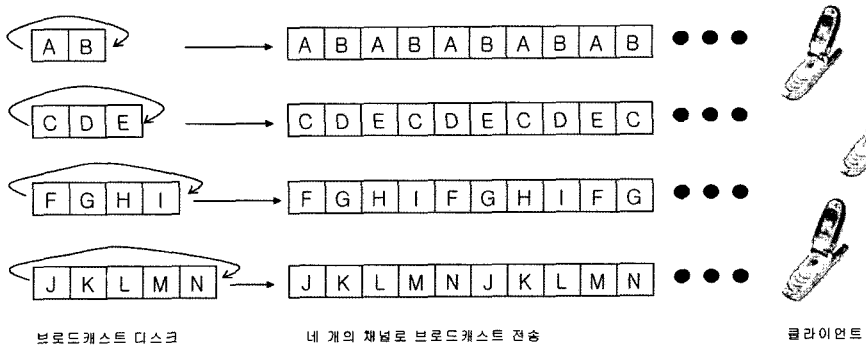

그림 1 브로드캐스트 시스템의 구조(K=3, N_1=2, N_2=3, N_3=4, N_4=5, N=14)

여 평균 기대 지연값(average expected delay)을 계산할 수 있다. 이를 위하여 데이타 아이템 d_i를 브로드캐스트 채널을 통해 얻기 위하여 기다려야 하는 시간의 기대치를 w_i라고 하자. 그러면 각각 p_i의 접근 확률(access probability)을 갖는 데이타 아이템들은 w_i의 순간만큼 지연이 되므로, 평균 기대 지연값(average expected delay)은 p_i와 w_i의 곱을 모든 i에 대하여 합한 값이 된다. 이것을 대수적으로 표현하면 다음과 같다.

평균 기대 지연값(average expected delay)= $\sum_{i=1}^{N} w_i p_i$ (1)

여기서 싱글 채널의 경우, 채널 하나를 통해 N개의 모든 데이타 아이템을 브로드캐스트 해야 하므로 모든 i에 대하여 $w_i = \frac{N}{2}$가 된다. 물론 이 값은 모든 데이타 아이템이 하나의 채널을 통해 모두 한 번씩만 할당되었다는 가정 하에서만 성립한다.

Greedy 알고리즘을 알아보기 위해 싱글 채널 평균 기대 지연값(Single Channel Average Expected Delay, SCAED)와 다중 채널 평균 기대 지연값(Multichannel Average Expected Delay, MCAED)에 대해서 설명하겠다. SCAED는 한 채널에 대한 평균 기대 지연값을 말한다. 한 채널에 데이타 아이템 i에서부터 j까지의 아이템이 포함되어 있는 경우의 평균 기대 지연값(Average Expected Delay)을 Cij라고 하면, Cij는 다음과 같다.

$$\text{SCAED} = C_{ij} = \left(\frac{j-i+1}{2}\right)\sum_{q=i}^{j} p_q,\ j \geq i, 1 \leq i, j \leq N \quad (2)$$

j-i+1은 한 채널에 포함된 데이타 아이템의 수를 의미한다. 이제 MCAED에 대하여 설명해 보도록 하자. 다중 채널(Multi-channel)을 통해 브로드캐스트 될 때, 채널 i에서 각각 N_i개만큼의 데이타 아이템을 브로드 캐스트 한다고 하면, 이 채널 i에 속한 모든 데이타 아이템 j에 대하여 기대 지연값(expected delay) wi는 $\frac{N_i}{2}$가 된다. 채널이 K개일 때, 평균 기대 지연값(average expected delay)은 다음과 같이 다시 나타낼 수 있다.

다중 채널 평균 기대 지연값(Multi-channel average expected delay) $= \frac{1}{2}\sum_{i=1}^{K}(N_i \sum_{d_j \in C_i} p_j)$ (3)

다중 채널을 가진 모바일 환경에서 데이타를 브로드캐스팅하기 위한 방법 중에 욕심쟁이(greedy) 방법은 수행시간이 빠르면서도 데이타의 접근 확률(access probability)을 잘 반영한 기법이다. 최적인 결과를 도출하는 동적 프로그래밍(Dynamic Programming)의 수행시간과 공간 복잡도(space complexity)에 대한 단점을 보완하면서 최적에 가까운 결과를 만들어 내는 방법이라고 할 수 있다. 욕심쟁이 방법이 어떠한 식으로 멀티플 채널 개수에 맞게 데이타 집합을 파티션 하는지 알아보기 위하여 간단한 예를 보자.

그림 2에서 보는 대로 Greedy 알고리즘은 후보자가 되는 각 분할 지점의 MCAED가 최소가 되는 지점을 최적의 분할 지점으로 선정한다. 이러한 분할 지점의 선정은 초기 파티션 수 1에서 시작하여, 주어진 채널의 수 K가 될 때까지 계속 반복 된다.

여기서 주시할 점은 반복을 시작하는 시점인 파티션 수, 즉 채널수가 1개 일 때는 어떠한 분할 지점도 만들지 않으므로 싱글 채널(single channel) 하에서는 Greedy 알고리즘 역시 FLAT 알고리즘과 같은 결과를 낸다는 것을 알 수 있다. 또한 특정 파티션 수에서 한 번 최적의 분할 지점이 선정되면, 이후에 그 분할 지점은 바뀌지 않으며, 최종 결과로 남게 된다. 이것이 최적의 결과를 만들어 내는 DP(Dynamic Programming) 알고리즘과 가장 큰 차이라고 할 수 있다.

Greedy 알고리즘의 슈도 코드(pseudo code)는 다음 그림 3과 같다.

동적 프로그래밍, DP 알고리즘과 VFK 알고리즘에 비하면 Greedy 알고리즘은 매우 빨리 수행된다. Greedy 알고리즘은 O((N+K)logK)의 시간 복잡도를 갖는

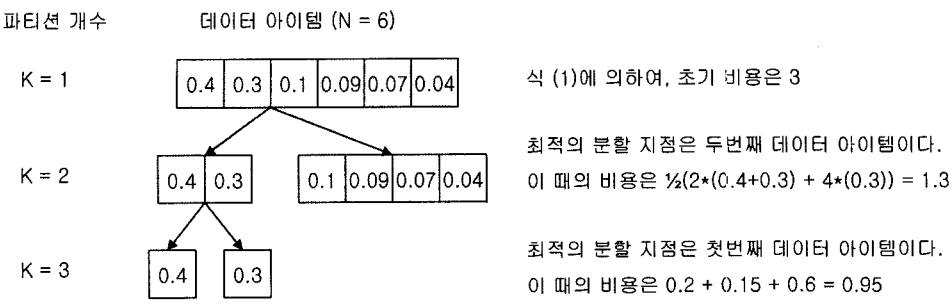

그림 2 Greedy 알고리즘의 예 (N=6, K=3)

```
input : set of N unit sized items ordered by popularity, K partitions
begin
        numPartitions := 1;
        while (numPartitions < K) {
                do Let point_c be the split point that most reduces
                        SCAED for each partition c.
                   Let point' be the point_c that most reduces MCAED.
                   Create a split at point'.
                   numPartitions := numPartitions + 1;
        }
end
```

그림 3 Greedy 알고리즘의 pseudo code

다. 이것은 지금까지 FLAT과 Bin-Packing 알고리즘에 비해서는 떨어지지만 VF^K보다는 나은 수행시간이다.

3. 다중 브로드캐스트 채널에서의 효과적인 데이타 할당

본 논문의 아이디어는 제일 먼저 모든 데이타에 대한 할당 횟수를 결정한 다음 이전 장에서 설명한 Greedy 방법으로 다중 채널로 데이타를 할당하고 이미 결정된 할당 횟수에 따라 데이타를 반복시키는 것이다. 이 방법은 단순히 Greedy방법으로 데이타를 다중채널에 할당하고 Acharya의 방법[3]인 채널 내에서 데이타의 접근 확률을 반영하여 데이타를 반복시키는 방법과는 달리 제일 처음 데이타에 대한 할당 횟수를 결정한다는 점에서 차이가 있다. 이것은 모든 데이타에 대한 전역적인 데이타 할당횟수를 반영했다는 점과 지역적으로 결정된 데이타의 반복횟수에 의해 데이타를 반복한다는 점에서 차이가 있다. 3.1장에서는 전역적인 데이타 반복횟수 결정을 위한 비용함수를 정의하고 3.2장에서는 전역적인 데이타의 반복 횟수를 결정짓는 방법에 대해서 설명하고 있다. 3.3장에서는 3.1장, 3.2장을 바탕으로 본 논문의 핵심인 S2AP알고리즘을 설명하고 있다.

3.1 비용 함수 정의

제안하는 데이타 할당 방법은 앞 장에서 설명한 브로드캐스트 시스템의 구조와 가정을 그대로 따른다. 그러나 이전 방법들에서 고려하지 않았던 동일 채널 내에서 데이타 아이템이 가지는 접근 확률(access probability)의 차이를 반영하기 위하여 새로운 변수를 첨가한다. 새로운 변수는 각 데이타가 각 채널의 전체 주기(major cycle) 내에서 할당되는 횟수이다. 기존의 MCAED에 이 변수를 사용하여 새로운 비용 함수를 정의한다. 이 비용 함수를 정의하기 위하여 다음과 같은 파라미터(parameter)를 정의한다.

N := the total number of Data Items.
K := the total number of channels.
$P := p_1, p_2, \cdots, p_N$ *where* $p_i \geq p_j$, for $1 \leq i \leq j \leq N$.
p_i := an access probability of a data d_i. $p_i \in \mathbb{R}$.
t_i := the number of a data d_i allocated in a major cycle.
$N_t = \sum_{i=1}^{N} t_i$

다음 정리 1과 정리 2를 통해 본 논문에서 제안하는 새 비용함수를 정의한다.

정리 1. 싱글 채널(single channel) 환경에서 평균 기대 지연, S2AP-SCAED은 $\frac{N_t}{2}\sum_{i=1}^{N}\frac{p_i}{t_i}$이다.

증명. 전제주기(major cycle)는 N_t이고, 데이타 아이템은 d_i는 전체 주기(major cycle) 내에서 t_i번 할당되므로 데이타 아이템 d_i에 대한 기대 지연(expected delay)은 $\frac{N_t}{2t_i}$이다. 또한 데이타 아이템 d_i는 p_i의 확률로 접근되므로 데이타 아이템 d_i에 대한 평균 기대 지연은 $\frac{N_t}{2t_i}\times p_i$ 이다. 따라서 전체 데이타 아이템에 대한 평균 기대 지연, S2AP-SCAED는 $\frac{N_t}{2t_1}p_1 + \frac{N_t}{2t_2}p_2 + \cdots + \frac{N_t}{2t_N}p_N$이다. 이것을 다시 쓰면, $\frac{N_t}{2}\sum_{i=1}^{N}\frac{p_i}{t_i}$ 이다. □

정리 2. 다중 채널(multiple channel) 환경에서 $N_i' = \sum_{d_j \in C_i} t_j$ 일 때, 평균 기대 지연, S2AP-MCAED는 $\frac{1}{2}\sum_{i=1}^{K}\left(N_i'\sum_{d_j \in C_i}\frac{p_j}{t_j}\right)$이다.

증명. 가정에 의하여 각 채널 i의 전체 주기(major cycle)는 N_i'이고, 그 채널 내에서 정리 1의 증명에서와 같이 데이타 아이템 dj의 기대 지연(expected delay)은 $\frac{N_i'}{2t_j}$이다. 정리 1의 증명에 의하여 채널 i에서의 전체 데이타 아이템에 대한 평균 기대 지연은 $\frac{N_i'}{2}\sum_{d_j\in C_i}\frac{p_j}{t_j}$이고, 이것을 모든 채널에 대하여 더한 값이 S2AP-MCAED가 된다. 즉 S2AP-MCAED는 $\frac{1}{2}\sum_{i=1}^{K}\left(N_i'\sum_{d_j\in C_i}\frac{p_j}{t_j}\right)$이 된다. □

만일 한 채널에 i부터 j까지의 데이타 아이템의 할당되어 있다면 그 채널에서의 평균 기대 지연(average expected delay)은 다음과 같다.

$$\text{S2AP Single Channel AED} = \frac{1}{2}\sum_{k=i}^{j}t_k\sum_{q=i}^{j}\frac{p_q}{t_q} \quad (4)$$

기존의 연구들은 싱글 채널 하에서 데이타 아이템의 할당이 오직 한 번씩만 일어나기 때문에 전체 주기(major cycle)의 길이가 곧 전체 데이타 아이템의 사이즈 N과 같았다. 하지만 제안하는 방법에서 정의한 비용함수(cost function)에서는 각 데이타들이 채널에 할당되는 횟수에 따라 전체 주기(major cycle)가 결정되기 때문에 각 데이타가 기다려야 하는 기대 지연(expected delay)이 N이 아닌 N_t가 된다. N_t는 모든 데이타 아이템을 할당한 후의 전체 주기(major cycle)이며, 이는 각 데이타 아이템의 할당 횟수 t_i의 합과 같다. 이것은 정리 1에서 보는 바와 같다.

정리 2의 S2AP-MCAED 역시 위에서 설명한 S2AP-AED와 같이 각 데이타의 할 당 횟수에 의하여 기대 지연(expected delay) N_i'이 결정된다. 즉 정리 2에서 보는 바와 같이 각 채널에서의 전체 주기(major cycle) N_i'는 그 채널에 할당된 데이타들의 할당 횟수 t_i의 총합과 같다.

3.2 각 데이타의 할당 횟수 결정 방법 및 알고리즘

2장에서 정의한 변수들은 브로드캐스트 환경에서 모두 주어지는 것들이다. 하지만 전체 주기(major cycle)에서 데이타 i가 할당되는 수 t_i는 그렇지 않다. 따라서 t_i를 결정할 필요가 있다. 이 장에서는 t_i를 결정하는 방법을 설명하도록 하겠다. t_i를 결정하기 위해서 정리 1을 이용한다. 정리 1은 싱글 채널에서의 평균 기대 지연(average expected delay)이고, 이 값을 최소로 할 수 있다면, 전체적인 성능의 향상이 생길 수 있다. 따라서 정리 1을 최소가 되게 하는 t_i를 선택하면 될 것이다. 이와 같은 의미에서 정리 1은 변수 N개의 t_i ($1\le i\le N$)에 대한 함수로 볼 수 있다. 따라서 정리 1을 다시 쓰면 다음과 같다.

$$f(t_1,t_2,\cdots,t_N)=\frac{N_t}{2}\left(\frac{p_1}{t_1}+\frac{p_2}{t_2}+\cdots+\frac{p_i}{t_i}+\cdots+\frac{p_N}{t_N}\right) \quad (5)$$

이제 식 (5)가 최소가 될 때의 t_i 값을 찾으면 된다. 이것은 다음 정리 3과 정리 4를 통해 구해진다.

정리 3. 함수 $f(t_1,t_2,\cdots,t_N)=\frac{N_t}{2}\left(\frac{p_1}{t_1}+\frac{p_2}{t_2}+\cdots+\frac{p_i}{t_i}+\cdots+\frac{p_N}{t_N}\right)$은 $\frac{N_t-t_1}{t_1}p_1=\frac{N_t-t_2}{t_2}p_2=\cdots=\frac{N_t-t_N}{t_N}p_N$ 일 때 최소값을 갖는다.

증명. $N_t=\sum_{i=1}^{N}t_i$ 이므로,

$$\begin{aligned}2f&=(t_1+t_2+\cdots+t_N)\left(\frac{p_1}{t_1}+\frac{p_2}{t_2}+\cdots+\frac{p_N}{t_N}\right)\\&=p_1+p_2+\cdots+p_N+\left(\frac{N_t-t_1}{t_1}p_1+\frac{N_t-t_2}{t_2}p_2+\cdots+\frac{N_t-t_N}{t_N}p_N\right)\end{aligned}$$

여기서 $\sum_{i=1}^{N}p_i=1$ 이므로,

$$2f-1=\left(\frac{N_t-t_1}{t_1}p_1+\frac{N_t-t_2}{t_2}p_2+\cdots+\frac{N_t-t_N}{t_N}p_N\right)$$

이다.

t_i는 자연수이므로, 변수 N개의 t_i에 대한 산술기하평균을 적용하면,

$$2f-1\ge N\left(\frac{N_t-t_1}{t_1}p_1\times\frac{N_t-t_2}{t_2}p_2\times\cdots\times\frac{N_t-t_N}{t_N}p_N\right)^{\frac{1}{N}}$$

이고, 이 때 등호는

$$\frac{N_t-t_1}{t_1}p_1=\frac{N_t-t_2}{t_2}p_2=\cdots=\frac{N_t-t_N}{t_N}p_N$$

일 때 성립한다.

따라서, 함수

$f(t_1,t_2,\cdots,t_N)=\frac{N_t}{2}\left(\frac{p_1}{t_1}+\frac{p_2}{t_2}+\cdots+\frac{p_i}{t_i}+\cdots+\frac{p_N}{t_N}\right)$는 $\frac{N_t-t_1}{t_1}p_1=\frac{N_t-t_2}{t_2}p_2=\cdots=\frac{N_t-t_N}{t_N}p_N$일 때 최소값을 갖는다. □

정리 3을 통해 함수 $f(t_1,t_2,\cdots,t_N)$가

$$\frac{N_t-t_1}{t_1}p_1=\frac{N_t-t_2}{t_2}p_2=\cdots=\frac{N_t-t_N}{t_N}p_N \quad (6)$$

일 때 최소가 됨을 보였다. 이제 정리 3을 이용하여 t_i를 구하면 된다. 정리 3에 의해서 얻어진 식 (6)를 통해 N개의 방정식을 만들어 낼 수 있고, 이것을 가지고 N

개의 변수 t_N에 대한 연립방정식으로 풀 수 있다. 그러나 연립방정식을 구하기 위해서는 $O(N^3)$의 시간 복잡도(Time Complexity)를 요구한다. t_i를 구하는 데에만 이렇게 많은 시간이 든다면 효과적일 수 없다. 따라서 시간을 줄일 수 있는 방법을 제안한다. 다음과 같이

$$\frac{N_t - t_i}{t_i} p_i = k \ , \ k \in \mathbb{R} \tag{7}$$

로 놓고, 실수 k의 값을 구할 수 있다면, 모든 t_i를 구할 수 있을 것이다. 정리 4를 통하여 O(N) 시간에 t_i를 구할 수 있다.

정리 4. $\frac{N_t - t_i}{t_i} p_i = k$ 라 하면, $\sum_{i=1}^{N} \frac{p_i}{k+p_i} = 1$ 이다.

증명. $\frac{N_t - t_i}{t_i} p_i = k$ 이므로, $t_i = \frac{p_i N_t}{k+p_i}$ 이다. 접근 확률(access probability)이 제일 작은 데이타 아이템은 당연히 최소로 할당될 것이다. $t_i (1 \le i \le N)$는 자연수이므로 접근 확률(access probability)이 제일 작은 t_N은 1이다. 따라서 $t_N = \frac{p_N N_t}{k+p_N} = 1$ 라 하면, $N_t = \frac{k+p_N}{p_N}$ 이다. 이것을 t_i에 대하여 정리하면, $t_i = \frac{p_i N_t}{k+p_i} = \frac{p_i (k+p_N)}{p_N (k+p_i)}$ 이 된다. 주어진 가정에서 모든 t_i의 합은 N_t와 같으므로, $N_t = \sum_{i=1}^{N} \frac{p_i (k+p_N)}{p_N (k+p_i)}$ 이다. 이 값은 $N_t = \frac{k+p_N}{p_N}$와 같은 값이므로 $\sum_{i=1}^{N} \frac{p_i (k+p_N)}{p_N (k+p_i)} = \frac{k+p_N}{p_N}$ 이고, 정리하면 $\sum_{i=1}^{N} \frac{p_i}{k+p_i} = 1$ 과 같다. □

정리 4를 통해 N개의 변수에 대한 방정식을 한 개의 변수 k에 대한 방정식으로 바꿀 수 있었다. 그러나 여전히 N개의 서로 다른 상수 $p_i (1 \le i \le N)$ 때문에, k를 p_i에 대한 일반식으로 나타내기는 힘들다. 따라서 방정식의 해를 구하는 방법 중의 하나인 뉴턴-랩슨 방법(Newton-Raphson method)을 사용할 것이다. 이 방법을 사용하기에 앞서 몇 가지 검토해야 할 것이 있다. 먼저, 위에서 만들어 낸 방정식 $\sum_{i=1}^{N} \frac{p_i}{k+p_i} = 1$이 우리가 찾고자 하는 범위 내에서 해를 갖는지의 여부와 그 해가 몇 개인지를 알아야 한다. $\frac{N_t - t_i}{t_i} p_i = k$ 라고 했을 때, 함수 f는 최소가 되고 그 값은 다음과 같다.

$$2f - 1 = N(k^N)^{\frac{1}{N}} = kN. \text{ 즉, } f = \frac{kN+1}{2} \tag{8}$$

식 (8)을 통해 알 수 있듯이 k가 1보다 커지면 기존의 평균 기대 지연(average expected delay)인 $\frac{N}{2}$보다 큰 값이 된다. 따라서 성능 향상을 꾀하려면 k가 1보다 작아야 함을 알 수 있다. 따라서 우리는 방정식 $\sum_{i=1}^{N} \frac{p_i}{k+p_i} = 1$이 0과 1 사이에서 해가 존재하는지의 여부와 몇 개의 해를 갖는지를 알아야 할 필요가 있다. 다음 정리 5는 이에 대한 답을 제공한다.

정리 5. 함수 $g(k) = \sum_{i=1}^{N} \frac{p_i}{k+p_i} - 1$ 는 0과 1사이에서 오직 하나의 해를 갖는다.

증명. $g(0) = \sum_{i=1}^{N} \frac{p_i}{p_i} - 1 = N - 1 \ge 0$, for $N \ge 1$.

$$\begin{aligned} g(1) &= \sum_{i=1}^{N} \frac{p_i}{1+p_i} - 1 = \frac{p_1}{1+p_1} + \frac{p_2}{1+p_2} + \cdots + \frac{p_N}{1+p_N} - 1 \\ &\le \frac{p_1}{1+p_n} + \frac{p_2}{1+p_n} + \cdots + \frac{p_N}{1+p_N} - 1 \\ &\qquad (\because p_i \ge p_N \text{ for } 1 < i < N) \\ &= \frac{1}{1+p_N}(p_1 + p_2 + \cdots + p_N) - 1 \\ &= \frac{1}{1+p_N} - 1 \qquad (\because \sum_{i=1}^{N} p_i = 1) \\ &= \frac{1-(1+p_N)}{1+p_N} = -\frac{p_N}{1+p_N} < 0 \qquad (\because p_N > 0) \end{aligned}$$

또한, k>0에서 g(k)의 도함수 g'(k)는

$$\begin{aligned} g'(k) &= \frac{-p_1}{(k+p_1)^2} + \frac{-p_2}{(k+p_2)^2} + \cdots + \frac{-p_N}{(k+p_N)^2} \\ &= -\sum_{i=1}^{N} \frac{p_i}{(k+p_i)^2} < 0 \end{aligned}$$

따라서 g(k)는 k>0인 구간에서 단조 감소 함수이다. 즉, $g(0) \ge 0, g(1) < 0$이고, k>0인 구간에서 단조 감소하므로 g(k)는 0과 1사이에서 오직 하나의 해를 갖는다. □

실제로 데이타 수에 따른 g(k)의 그래프를 그려보면 k>0인 구간에서 g(k)가 1보다 작은 지점에서 해를 갖

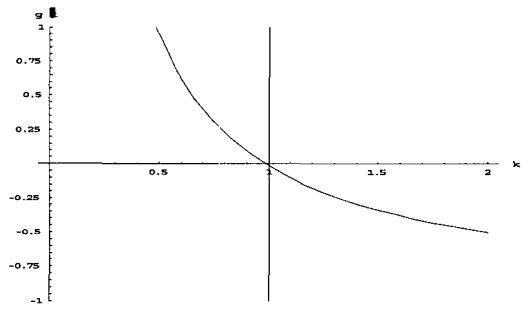

그림 4 함수 g(k)의 그래프. N=1000 (구간 k∈[0, 2])

는다는 것을 알 수 있다. 그림 4는 N=1000일 때, g(k)의 그래프이다.

지금까지 각 데이타의 할당 횟수 t_i를 결정하기 위한 방법을 설명하였다. k에 대한 함수 g(k)의 해를 구할 수 있다면, k값을 정할 수 있고 이것을 가지고 모든 t_i를 구할 수 있을 것이다. g(k)의 해는 뉴턴-랩슨 법을 통해 구한다. 뉴턴-랩슨 법(Newton-Raphson Method)은 함수의 해를 구하는 방법 중에서 가장 많이 사용되는 방법 중 하나이다. 이 방법을 간단히 설명하면, 임의의 x_0값에서 시작하여 $f(x_0)$에서의 접선이 x축과 만나는 점을 x_1로 놓는다. 다시 $f(x_1)$에서의 접선이 x축과 만나는 점을 x_2라 하고, 다시 $f(x_2)$에서의 접선과 x축의 교점을 찾는다. 이와 같은 방법을 반복하면 f(x)=0인 값, 즉 함수 f의 해에 수렴하게 된다[8]. 종료 판정 기준을 정하기 위하여, 이미 보장된 허용오차 ε_s를 사용한다. $\epsilon_s=(0.5\times 10^{2-n})\%$이 기준이 만족된다면, 결과가 적어도 n개의 유효숫자에 대해서는 정확하다는 것을 보장한다[8]. 따라서 미리 정해 놓은 오차에 대하여, 뉴턴-랩슨 공식(Newton-Raphson equation)을 통해 얻은 $g(x_i)$가 허용오차보다 작을 때 종료한다면, 원하는 오차 범위 내에서의 해를 얻게 된다. 뉴턴-랩슨 법을 통하여 g(k)=0 일 때, k의 값을 구할 수 있다. 이제 이 k의 값을 통하여 각 데이타 아이템의 할당 횟수 t_i를 구하는 방법과 알고리즘을 설명하도록 하겠다.

정리 6. $\frac{N_t-t_i}{t_i}p_i=k$일 때, 다음 평균 기대 지연 S2AP-SCAED는

$$f(t_1,t_2,\cdots,t_N)=\frac{N_t}{2}\left(\frac{p_1}{t_1}+\frac{p_2}{t_2}+\cdots+\frac{p_i}{t_i}+\cdots+\frac{p_N}{t_N}\right)$$

을 최소로 하는 자연수 t_i는 $\frac{p_i(k+p_N)}{p_N(k+p_i)}$ 을 반올림한 수이고, $\partial\left[\frac{p_i(k+p_N)}{p_N(k+p_i)}\right]$으로 표기한다.

증명. 가정에 의하여, $\frac{N_t-t_i}{t_i}p_i=k$, $k\in\mathbb{R}$이다. t_i에 대해서 정리하면, $t_i=\frac{p_iN_t}{k+p_i}$이다. 정리 4의 증명에서 다음을 얻을 수 있다. $t_i=\frac{p_iN_t}{k+p_i}=\frac{p_i(k+p_N)}{p_N(k+p_i)}$. 이 식에서 k와 p_i, p_N는 실수이므로 t_i 역시 실수이다. 하지만, 데이타를 브로드 캐스트 채널에 실수만큼 할당할 수 없으므로 여기서 약간의 근사가 필요하다. 정리 3에서 $\frac{N_t-t_1}{t_1}p_1=\frac{N_t-t_2}{t_2}p_2=\cdots=\frac{N_t-t_N}{t_N}p_N$일 때, 평균 기대 지연이 최소가 된다는 것을 보였다. 여기서, $\frac{N_t-t_1}{t_1}p_1=\frac{N_t-t_2}{t_2}p_2=\cdots=\frac{N_t-t_N}{t_N}p_N$라는 의미는 산술기하의 측면에서 이 값들이 서로 최대한 가까이 있을 때임을 뜻한다. 따라서 주어진 평균 기대 지연(average expected delay)을 최소로 하는 자연수 t_i는 $\frac{p_i(k+p_N)}{p_N(k+p_i)}$ 값에 가장 가까운 정수인 $\partial\left[\frac{p_i(k+p_N)}{p_N(k+p_i)}\right]$이다. □

앞에서 설명한 방법을 토대로 그림 5에서는 할당 횟수 t_i를 결정하는 알고리즘을 보여주고 있다.

예제 1. p_1=0.45, p_2=0.19, p_3=0.17, p_4=0.08, p_5=0.07, p_6=0.06, p_7=0.06이고, 앞에서 설명한 바와 같이 앞에서 설명한 바와 같이 유효 수자 10자리의 정확도를 갖도록 오차를 허용하면, 허용 오차는 $\epsilon_s=(0.5\times 10^{2-10})\%$이 된다. 그림 5의 알고리즘을 통해 실제 해를 구해보면, 채널이 하나일 때, 즉 파티션이 필요 없을 때, start=1이고 end=7이 된다. 이 때 뉴턴-랩슨 법 부분의 알고리즘에 의하여 k=0.7760684799 이다. 이 k 값이 다음 코드에서 t_1부터 t_7까지 구하는데 사용된다. 실제로 구해보면 다음과 같다. 식 $t_i=\partial\left[\frac{p_i(k+p_N)}{p_N(k+p_i)}\right]$ 에 p_i와 k값을 각각 대입하면, t_1=5, t_2=2, t_3=2, t_4=1, t_5=1, t_6=1, t_7=1. 이 때 분명히 $\sum_{j=1}^{7}p_j=1$ 이다. 채널이 2인 상황, 즉 한 번의 분할을 할 때(d_1, d_2, d_3)와 (d_4, d_5, d_6, d_7)로 분할되고, 이 때 각 채널 내에서 데이타 아이템의 할당 횟수는 다음과 같다. 채널 1에서 t_1=2, t_2=1, t_3=1이고 채널 2에서 t_4=1, t_5=1, t_6=1, t_7=1이 된다.

위의 예제 1에서 보는 것과 같이 그림 5의 할당 횟수 결정 알고리즘은 데이타 접근확률과 채널의 개수에 따라 최적의 할당 횟수를 결정할 수 있다. 우선 데이타 접근 확률이 높은 데이타에 대해서는 그렇지 못한 데이타에 비하여 더 높은 할당횟수를 갖는다. 그러나 예제 1에서 확인한 것과 같이 단일 채널일 때와는 다르게 채널 개수가 증가함에 따라 각 채널 내 데이타들끼리의 상대적인 접근 확률이 적용되는 것을 확인할 수 있다. 그러므로 그림 5의 할당 횟수 결정 알고리즘은 채널 개수와 데이타 접근 확률의 변화에 따라 유동적으로 변할 수 있는 것을 확인할 수 있다.

3.3 S2AP 알고리즘

이 장에서 제안할 S2AP 알고리즘은 기본적으로 Greedy 방법을 사용한다. 또한 기존 Greedy 알고리즘에서는 없었던 동일 채널 내에 전체 주기(major cycle) 안에서의 데이타 아이템의 할당 횟수를 반영하기 위하여 [3]에서 제시된 브로드캐스트 프로그램 생성 기법을 이용하기로 한다. 알고리즘의 설명을 위하여 이것을 균

```
Input: P={p1,p2,···, pN} ordered by access probability,
   start := start split point, end := end split point, ε_s := acceptable error.
Output:  t_i := the number of a data i allocated in major cycle. t_i ∈ ℕ.
begin
        k=c; /* starting in a arbitrary constant. */
        while(g(k, start, end)<ε_s) {  /* Newton-Raphson method */
                k = k - g(k, start, end)/g'(k, start, end);
        }
        for(i=start; i<=end; i++) { /* Theorem 6 */
                t_i = round[ p_i(k+p_end) / p_end(k+p_i) ];
        }
end
function g(k, start, end) {  /* g(k) in Theorem 5 */
        float result = 0.0;
        for(i=start; i<=end; i++) {
                result = p_i/(k+p_i);
        }
        return result - 1;
}
function g'(k, start, end) { /* the differential of g(k) in Theorem 5 */
        float result = 0.0;
        for(i=start; i<=end; i++) {
                result = p_i/(k+p_i)^2;
        }
        return (-result);
}
```

그림 5 할당 횟수 결정 알고리즘

형 할당(BA: balance allocation)이라 하자. 다음 예제 2는 BA가 데이타 아이템을 채널에 어떻게 할당하는지를 보여준다.

예제 2. 데이타 아이템의 수 N=11이다. 각 데이타 아이템의 할당 횟수 $t_1=4$, $t_2=t_3=2$, $t_4=t_5=t_6=t_7=t_8=t_9=t_{10}=t_{11}=1$ 이다. 이 때 균형 할당(balance allocation)은 그림 6과 같이 데이타를 할당한다.

이제 본 논문에서 제안하는 S2AP 알고리즘의 커다란 구성은 크게 두 가지 단계로 나누어진다. 먼저 3.2장에서 설명한 할당 횟수 결정 알고리즘을 통해 각 파티션에서의 t_i를 결정한다. t_i가 결정되면 각 분할 지점(split point) 중에서 식 (3)을 최소로 하는 지점에서 채널 파티션을 한다. 파티션이 이루진 후에는 해당 채널 내에 균형 할당(balance allocation)방법으로 데이타를 할당하게 된다. 이 데이타 할당에 대한 반복은 채널 수 K만큼 계속된다. 다음 그림 7은 제안하는 S2AP 알고리즘을

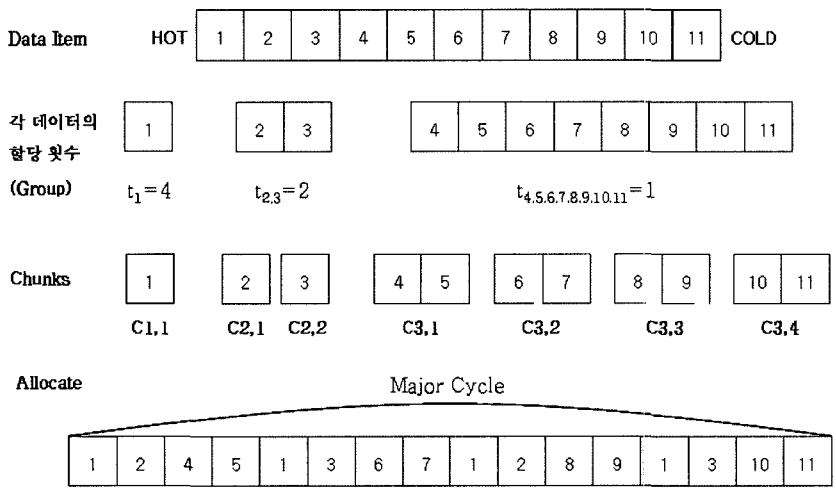

그림 6 균형 할당(balance allocation)의 예

```
Input: P={p1,p2,···, pN} ordered by access probability, K partitions.
Output: broadcast program of data items allocated.
Phase 1. <Partition phase>
    partitions = 1;
    Determine t1~tN in each partition by Algorithm in Figure 5;
    while (partitions < K) {
        for(i=1; i<=N-1; i++) {
            cand_split[i] = S2AP-MCAED of a every candidate for the split point i;
        }
    split[partitions] = MIN_{i∈N}{cand_split[i]}
    Create a split at split[partitions];
        Sort split[partitions] for the number of partitions.
        partitions++;
    }
Phase 2. <Data allocation phase in each channel by using BA.>
    partitions = 1;
    while (partitions < K) {
        N'[partitions] = the number of data items in each channel;
        for(i=1; i<=N'[partitions]; i++) {
            alloc_num(i) = ti;
        }
        grouping by same alloc_numi;
        total_group = the number of groups in each channel;
        for(j=1; j<=total_group; j++) {
            group_num(j) = the number of data items in same group;
        }
        for(i=1; i<=total_group; i++){
           rel_rep(i)=alloc_numi;
        }
        Split each group into a number of smaller units. These units are called
        chunks(Cij refer to the jth chunk in groupi).
        max_chunks = the Least Common Multiple(LCM) of the relative repeat(rel_rep).
        for( i=1; i<=total_group; i++){
            num_chunks(i)=max_chunks/rel_rep(i) chunks.
        }
        Create the broadcast program by interleaving the chunks of each disk in the
        following manner :
        for(i=0; i<max_chunks; i++) {
            for(j=1; j<group_num; j++) {
                Broadcast chunks C_{j,(i mod num_chunks(j))}
            }
        }
        partitions++;
    }
```

그림 7 S2AP 알고리즘

보여주고 있다.

우선 모든 데이타 N개의 접근 확률분포와 채널의 개수 K를 입력 값으로 넣어준다. 파티션 단계인 1단계에서는 그림 5의 알고리즘을 통해 각 데이타 마다 한 major cycle동안 반복될 횟수 $t_i(1 \le i \le N)$를 결정한다. 이렇게 결정된 t_i를 이용하여 모든 데이타에 대하여 정리 2의 S2AP-MCAED를 계산하여 S2AP-MCAED, 즉 평균지연시간이 가장 작아지는 지점으로 파티션으로 분할할 시점을 결정한다. 위의 파티션 단계가 Greedy방법과 구별되는 부분은 Greedy방법에서는 MCAED를 최소화하는 시점을 채널 파티션의 시점으로 결정하였지만 S2AP방법에서는 기존의 MCAED에 각 데이타의 할당 횟수 t_i를 반영한 평균지연 시간인 S2AP-MCAED를 최소로 하는 시점을 파티션 분할의 지점으로 한다는 것이다.

각각의 채널내의 데이타 할당단계인 2단계에서는 특

정 채널에 할당된 데이타의 반복횟수를 alloc_num이라고 하고 동일한 alloc_num에 따라 그룹을 짓는다. 특정 채널에서 동일한 반복횟수에 따라 그룹 지어진 데이타의 집합의 개수를 total_group이라고 한다. 또한 동일한 그룹에 있는 데이타 아이템의 개수를 group_num이라고 한다. 만약 예제 2와 같이 t_i값이 결정되었다면 group1에 할당된 데이타 아이템은 d1이고 alloc_num은 4, group_num은 1이다. 같은 방식으로 group 2에 할당된 데이타 아이템은 d2,d3이고 alloc_num은 2, group_num은 2이다. 또한 group3은 데이타 아이템 d4,d5,d6,d7, d8,d9,d10,d11이고 alloc_num은 1, group_num은 8이다. 따라서 rel_rep(1)=4, rel_rep(2)=2,rel_rep(3)=1이 된다. 이제 알고리즘에 따라 그룹을 더작은 단위인 chunk로 나누게 된다. max_chunks의 값이 4이므로 num_chunks(1)=1, num_chunks(2)=2, num_chunks(3)=4가 되고 실제 데이타는 relative repeat의 최소 공배수 값이 4이므로 4개의 minor cycle로 브로캐스팅된다.

예제 3. 기존 연구는 데이타 집합을 파티션하기 때문에 동일 채널 내에서 각 데이타는 오직 한 번씩만 할당된다. 그림 8은 이와 같은 식으로 할당될 수 있는 한 예를 보여주고 있다. 그림 8과 같은 상황에서 SCAED와 MCAED를 계산해 보면 다음과 같다.

$$\text{channel 1> } AED = \frac{7}{2} = 3.5$$

그림 8과 같이 채널이 두 개와 세 개일 때, 멀티채널 평균 기대 지연은 다음과 같다.

$$\text{channel 2> } MCAED = \frac{1}{2}(2\times 0.8 + 5\times 0.2) = 1.3$$

$$\text{channel 3> } MCAED = \frac{1}{2}(0.6 + 0.2 + 5\times 0.2) = 0.9$$

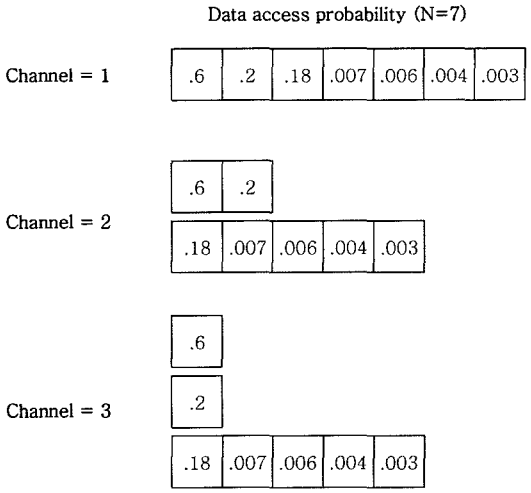

그림 8 기존 알고리즘으로 할당할 수 있는 예

예제 4. 그림 9는 N=7, K=3인 상황에서 제안된 알고리즘으로 할당된 예를 보여주고 있다. 그림 9와 같이 제안된 방법을 사용하면 동일 채널 내에서도 접근 확률(access probability)이 높은 데이타 아이템을 여러 번 할당할 수 있다. 이 상황에서 싱글 채널일 때 average expected delay는 다음과 같다.

$$\text{channel 1> } AED = \frac{12}{2}\left(\frac{0.6}{4} + \frac{0.2}{2} + \frac{0.18}{2} + 0.007 + 0.006 + 0.004 + 0.003\right)$$
$$= 6(0.15 + 0.1 + 0.09 + 0.02)$$
$$= 6(0.36) = 2.16$$

그림 9와 같은 두 개 채널을 가정하면, 멀티채널 average expected delay는 다음과 같다.

$$\text{channel 2> } MCAED = \frac{1}{2}\left(1\times 0.6 + 8\times\left(\frac{0.2}{2} + \frac{0.18}{2} + 0.007 + 0.006 + 0.004 + 0.003\right)\right)$$
$$= \frac{1}{2}(0.6 + 8(0.1 + 0.09 + 0.02)$$
$$= \frac{1}{2}(0.6 + 1.68) = 1.14$$

$$\text{channel 3> } MCAED = \frac{1}{2}\left(0.6 + 0.2 + 6\times\left(\frac{0.18}{2} + 0.02\right)\right)$$
$$= \frac{1}{2}(0.8 + 0.66) = 0.73$$

예제 3과 예제 4에서 보듯이 동일 채널 내에서도 접근 확률(access probability)이 높은 데이타 아이템을 여러 번 할당하게 되면, 평균 접근 시간을 줄일 수 있다는 것을 알 수 있다. 이를 통하여 전체적인 시스템 성능의 향상을 꾀할 수 있을 것이다.

기존의 Greedy 알고리즘에 비하여, 제안하는 S2AP 알고리즘은 채널내의 데이타 할당 횟수 k를 구하기 위해 더 많은 시간이 필요하다. 허용오차를 소수점 이하 L자리까지 할 경우, 각 분할지점에서 k를 구하는데

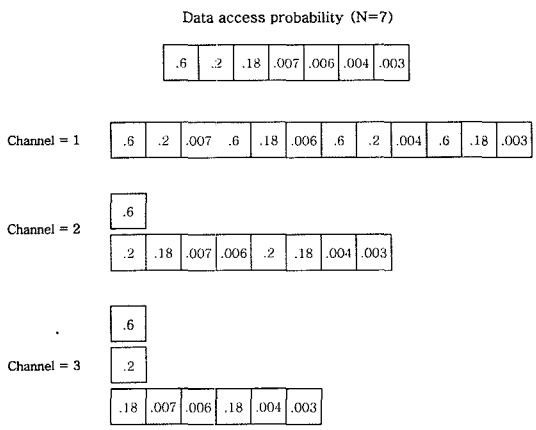

그림 9 Example of data allocation

$O(N\log_2 L)$만큼의 시간이 더 필요하다. 따라서 기존 Greedy 알고리즘의 시간 복잡도(Time Complexity) $O((N+K)\log K)$에 곱하면, 제안하는 S2AP 알고리즘의 시간 복잡도(Time Complexity)는 $O(N(N+K)\log K\log L)$가 된다. 이것은 $O(N^2)$ 보다는 크고, $O(N^3)$보다는 작음을 알 수 있다.

4. 성능 분석 및 결과

이 장에서는 앞의 2장 관련 연구에서 언급했던 네 가지 연구와의 성능을 비교할 것이다. 4.1장에서는 이 장에서의 실험 환경과 변수에 대하여 언급할 것이다. 4.2장에서는 각 알고리즘이 최종 결과를 내는 데까지 걸리는 수행시간을 비교할 것이며, 4.3장에서는 특정 상황에서의 평균 접근 시간에 대해서 분석할 것이다. 이장의 표와 그래프에서의 각 알고리즘의 표기는 다음 표 2와 같다.

표 2 성능 분석을 위해 비교되는 알고리즘

알고리즘	표기
FLAT [3]	flat
Bin-packing [5]	bp
VF^K [6]	vfk
GREEDY [4]	greedy
제안하는 알고리즘	s2ap
비교 알고리즘	greedy + BA

또한 이 실험에서 사용할 데이타의 접근확률은 지프 분포(Zipfian distribution)를 따른다. 실제 모바일 환경에서의 데이타에 대한 선호도는 이 분포를 따른다고 알려져 있다. 지프 팩터(zipf factor)로 불리는 θ값은 지프(zipf) 분포의 치우침 정도를 나타낸다. $\theta=0$일 때, 모든 데이타는 같은 접근 확률을 갖게 되고 지프 분포는 유니폼 분포(uniform distribution)가 된다. 이 θ값이 커질수록 지프 분포는 더욱 치우쳐진 분포(skewer distribution)가 되는데, 이것은 클라이언트들이 특정 데이타들에 대한 선호가 높고 대부분의 나머지 데이타에 대해서는 접근 확률이 매우 낮아짐을 의미한다.

전역적인 할당 횟수를 구하는 것을 특징으로 하는 제안하는 알고리즘의 성능 비교를 위해 Greedy 방법으로 전체 데이타를 각 다중채널에 할당하고 각 채널의 할당된 데이타를 다시 balanced allocation 하는 비교 알고리즘을 추가하였다.

4.1 실험 환경

본 논문의 실험 환경은 표 3과 같다. 각 알고리즘의 구현은 각 논문에서 제시한 슈도 코드(pseudo code) 또는 알고리즘 설명에 근거하여 C언어로 구현하였다.

성능 비교는 브로드 캐스트 해야 할 전체 데이타의 사이즈, 채널의 수, 그리고 데이타 접근 확률의 치우침 정도를 나타내는 지프 팩터(zipfian factor)가 변화함에 따라 알고리즘의 성능이 어떻게 변화하는지를 알아볼 것이다. 표 4는 각 변수 값들의 기본 값과 그 범위를 나타내고 있다.

표 3 실험 환경

CPU	인텔 펜티엄 4 프로세서 2.4GHz
Memory	512MB RAM
OS	Microsoft Windows XP Professional

표 4 성능 분석을 위한 변수

변수	범위	기본값	변화량
N : 데이타의 수	(500, 5000)	2000	500
θ : Zipfian factor	(0, 1.0)	0.95	0.1
K : 물리 채널 수	(1, 10) (100,1000)	4	1 100
simulation 횟수		20	

앞으로의 실험은 각 실험에서 변화를 주어야 하는 변수와 변화를 주어야 하는 변수 이외의 고정되야 하는 기본 변수 모두 위의 표 4를 바탕으로 할 것이다.

4.2 수행 시간(Execution Time)

이 장에서는 앞에서 설명한 알고리즘의 수행시간을 비교할 것이다. 그림 10은 수행시간에 대한 실험결과를 보여주고 있다. 실험은 표 4의 기본 값을 따르고 데이타의 수를 500에서 5000까지 변화시키면서 수행시간을 측정하였다.

FLAT 알고리즘을 제외하고, 모든 알고리즘은 데이타 사이즈가 커짐에 따라 수행시간이 증가하였다. FLAT 알고리즘은 2.1장에서도 설명하였듯이 시간 복잡도가 데이타의 수 N과는 무관하고, 채널 수 K에 종속된 O(K)이다. 따라서 그림 10과 같은 결과가 나온다. 본 논문에서 제안하는 S2AP 알고리즘의 수행시간이 가장 크게 나오는데, 그것은 앞에서도 설명했듯이 각 데이타의 할당 횟수 t_i를 결정하기 위해 추가적인 시간이 필요하기 때문이다. 반면에 VF^K 역시 본 논문에서 제안하는 S2AP 알고리즘에 가까운 수행시간을 보인다. 그것은 앞의 2.4장에서 설명한 것처럼 VF^K 알고리즘 역시 상당히 높은 시간 복잡도(Time Complexity) $O(KN^2\log K)$를 갖기 때문이다. 이에 반해 Greedy 알고리즘과 Bin-packing 알고리즘의 수행시간은 이에 비해 낮다는 것을 알 수 있다. 비교 알고리즘인 Greedy + BA 알고리즘은 Greedy 알고리즘보다 약간의 긴 수행시간을 보이고 있다. 이것은 제안하는 알고리즘의 각 데이타의 할당 횟수를 전역적으로 결정하는 데 반해 Greed + BA 알고리즘에는 각 채널 내에서 지역적으로 데이타의 할당 횟수를 결정하기 때문이다.

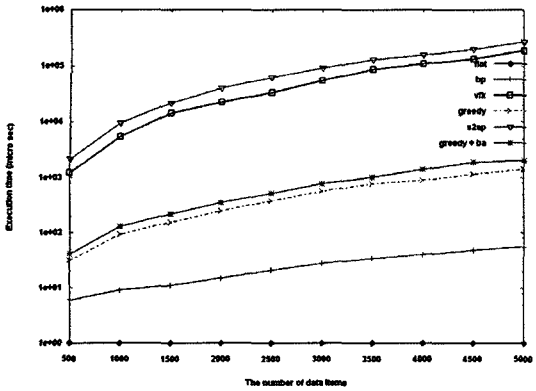

그림 10 각 알고리즘의 수행시간

4.3 평균 접근 시간(Average Access Time)

수행 시간에 대한 성능 비교에 더하여 평균 접근 시간에 대한 성능을 비교하고자 한다. 평균 접근 시간(average access time)에 대한 측정은 알고리즘 적용 시에 사용한 지프 팩터(zipfian factor) 값과 같은 지프 분포에 따라 데이타를 얻어 내어 측정하였다.

4.3.1 채널수에 따른 평균 접근 시간

본 장에서는 채널수를 1에서 10까지, 100에서 1000까지 증가시킴에 따른 평균접근시간을 측정해 보았다 . 그림 11에서 보면, FLAT 기법의 평균 접근 시간(average access time)이 전 채널에 걸쳐 제일 크다는 것을 알 수 있다. 바로 밑으로 Bin-packing 알고리즘이 위치하고 있다. FLAT 보다는 좋은 성능을 보이지만 다른 알고리즘 보다는 평균 접근 시간이 높게 나오고 있다. VF^K와 GREEDY 알고리즘은 거의 비슷하거나 근소하게 VF^K 알고리즘이 좋은 성능을 내고 있으며, 본 논문에서 제시한 S2AP 기법은 채널이 많아질수록 그 차이가 줄어들기는 하지만, 전반적으로 제일 작은 평균 접근 시간을 보이고 있다.

그림 11에서의 큰 특징 중 하나가 채널이 하나일 때의 결과인데, 본 논문에서 제안한 알고리즘을 제외한 모든 알고리즘이 FLAT과 동일한 평균 접근 시간을 보이고 있는 것이다. 이것은 기존 멀티채널에 대한 할당 알고리즘이 싱글 채널의 상황을 고려하지 않았기 때문이며, 데이타 집합의 파티션에 그 목적을 두었기 때문에 나타나는 현상이다. 즉 기존 멀티채널을 위한 할당 알고리즘은 싱글 채널이 되면 FLAT과 같은 결과를 만들어 내게 된다. 반면에 본 논문에서 제시한 S2AP 알고리즘은 채널이 하나일 때도 할당 횟수를 반영하기 때문에 다른 알고리즘과 비교할 때 성능이 월등하다. 기존 알고리즘과 제안한 S2AP 알고리즘의 평균 접근 시간의 차이가 채널의 수가 많아질수록 줄어드는데, 이 이유는 채널이 많아질수록 기존 알고리즘 역시 더욱 세밀한 파티션이 생겨서 동일 채널 내에서 데이타의 접근 확률의 차이가 작아지기 때문이다. 하지만 채널수가 비교적 작은 4이하인 경우 기존 알고리즘에 비해 성능이 상당히 우수하다는 것을 알 수 있다. Greedy + BA 알고리즘의 경우에는 GREEDY 알고리즘과 VF^k의 중간 성능을 보이며 접근 시간에 있어서 확연한 성능의 우수성을 보이지 못하고 있다. 이것으로 S2AP 알고리즘이 전역적으로 데이타 할당 횟수를 결정하는 데 반해 GREEDY + BA 알고리즘은 지역적으로 데이타 할당 횟수를 결정하기 때문에 전역적인 데이타 할당 방법이 접근 시간에 있어 우월하다는 사실을 알 수 있다. 채널수를 100개에서 1000개로 급격히 증가시킨 경우에도 제안한 S2AP알고리즘은 지속적으로 좋은 성능을 보이고 있음을 그림 11의 두 번째에서 확인할 수 있다.

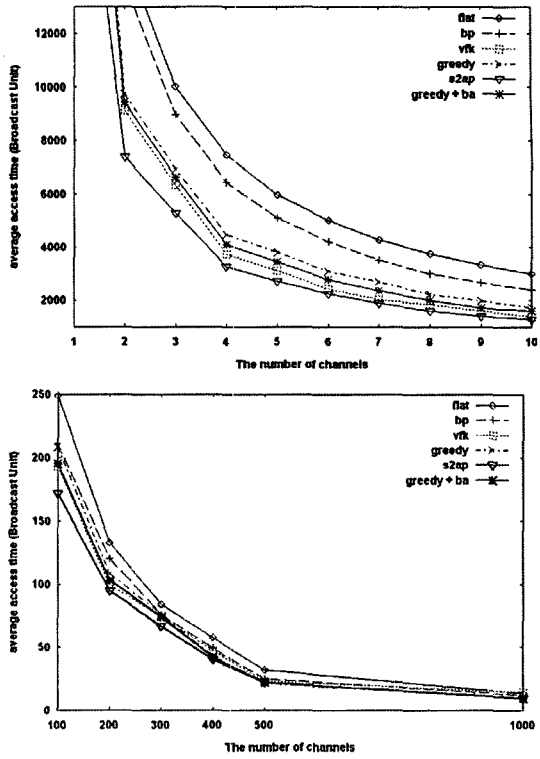

그림 11 채널 수 변화에 따른 평균 접근 시간(θ = 0.95)

그림 12는 지프 팩터 θ를 0.55로 고정시키고, 채널수를 100개에서 1000개로 변화시키며 각 알고리즘들의 평균 접근 시간을 측정한 그림이다. 그림 11과의 차이점은 채널 개수 200개를 넘어서면서 가장 긴 평균 접근 시간을 보이는 것은 VF^K 알고리즘이다. 또한 BP 알고리즘도 FLAT 알고리즘보다 긴 평균 접근 시간을 나타내고 있다. 하지만 S2AP 알고리즘은 채널 개수의 변화에도 여전히 가장 좋은 평균 접근 시간을 보이고 있다. 그러

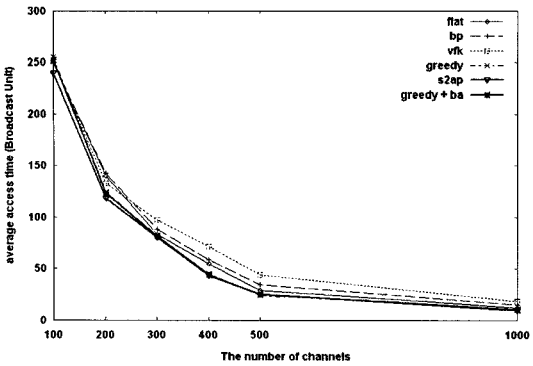

그림 12 채널 수 변화에 따른 평균 접근 시간($\theta = 0.55$)

나 채널 개수가 급격히 증가하면서는 VF^K와 BP 이외의 알고리즘과는 큰 성능 차이를 보이지는 않고 있다.

4.3.2 데이타 수에 따른 평균 접근 시간

이번 실험에서는 기본 채널수와 지프팩터 값을 사용하고, 데이타의 수를 500개에서 5000개까지 변화시키면서 평균 접근 시간(average access time)을 측정하였다. 그 결과는 그림 13과 같다.

제일 상위에 FLAT 기법이 위치한다. 그 밑으로 Bin-packing 알고리즘이 있고, VF^K와 GREEDY, 비교 알고리즘이 근소한 성능을 보이며 위치하고 있다. 본 논문에서 제시한 S2AP 알고리즘의 평균 접근 시간이 좋음을 알 수 있고, 그래프의 기울기가 상대적으로 작은 것으로 보아 데이타 수가 많아질수록 성능의 차이는 커질 것이라는 것을 예측할 수 있다. 이와 같이 데이타 수가 커짐에 따라 S2AP의 평균접근시간이 다른 알고리즘에 비해 작은 이유는 다음과 같다. 채널의 개수가 고정된 상태에서 데이타의 개수가 증가하면 특정 채널에서 방송해야 하는 데이타의 개수는 점점 증가할 것이다. 이때 다른 다중채널 파티션 알고리즘은 특정 채널안의 데이타들을 데이타 접근분포에 상관없이 방송하여 데이타 수의 증가에 따른 평균 접근 시간이 증가하는 반면 S2AP는 특정채널 안에서 데이타 접근분포를 반영하여 인기 있는 데이타를 더 자주 방송하기 때문에 다른 다중채널 파티션 알고리즘에 비해 평균 접근시간이 감소한 것이다.

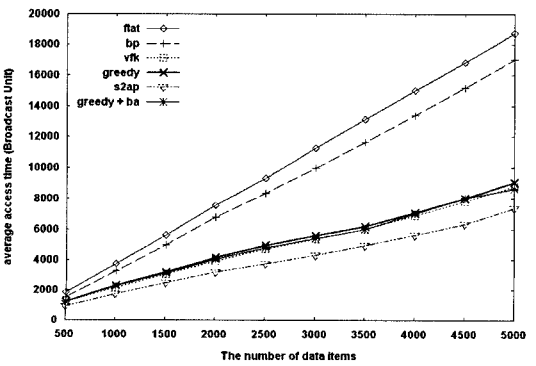

그림 13 데이타 수의 변화에 따른 평균 접근 시간

4.3.3 지프 변화에 따른 평균 접근 시간

이번 실험에서는 기본 데이타의 수와 채널의 수를 사용하고, 데이타의 접근 확률 분포에 대한 변화, 즉 지프 팩터 θ를 변화시키면서 평균 접근 시간을 측정하였다. 그림 14에서 그 결과를 보여주고 있다.

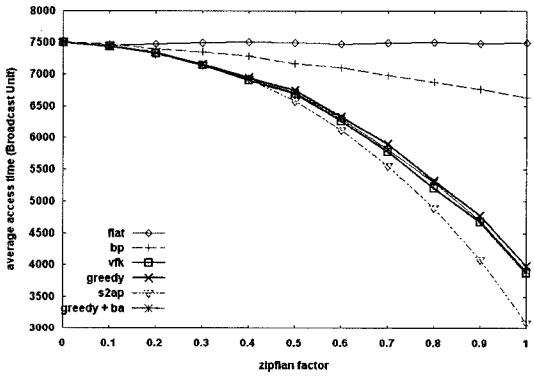

그림 14 지프 팩터 변화에 따른 평균 접근 시간

데이타의 접근 확률을 전혀 반영하지 않은 FLAT 기법은 지프 팩터(zipfian factor) 값의 변화에 상관없이 일정한 결과를 보이고 있다. 다음으로 Bin-packing 알고리즘이 있고, Greedy와 비교 알고리즘, VF^K 알고리즘의 차례로 위치한다. 또한 지프 팩터 값이 0일 때, 즉 데이타의 접근 확률이 모두 같은 유니폼 분포(uniform distribution)일 때는 모든 알고리즘이 같은 평균 접근 시간(average access time)을 갖는다는 것을 알 수 있다. 지프 팩터 값의 변화에 따른 평균 접근 시간 역시 본 논문에서 제시한 S2AP 알고리즘이 가장 나은 성능을 보이고 있다. 이와 같이 S2AP가 가장 적은 평균접근 시간을 보인 이유는 다른 다중채널 파티션 알고리즘들은 지프 변화에 따라 응답시간을 줄이기 위한 채널 파티션만을 고려했지만 S2AP는 채널 파티션뿐만 아니라 분할된 채널 내에서도 지프 변화에 따라 데이타가 자주 방송되기 때문이다.

5. 결 론

본 논문에서는 다중 채널 환경에서 브로드캐스트를 위한 데이타 할당 방법을 제안하였다. 기존의 다중 채널 환경에서의 데이타 할당 방법들은 데이타 집합을 채널 수에 맞게 분할하는 문제로 생각하였다. 이러한 문제로 보면 동적 프로그래밍(Dynamic Programming, DP) 알

고리즘이 최적의 결과를 만들어 낸다. 즉 동적 프로그래밍 알고리즘을 이상의 성능 향상을 이루어 낼 수는 없었다. 그래서 최적의 결과에 근접한 결과를 만들어 내면서 수행시간이 빠른 알고리즘이 고안되었다.

하지만 다중 채널 환경에서 브로드캐스트를 위한 데이타 할당 방법을 단순히 데이타 집합의 파티션으로 보지 않는다면, 평균 접근 시간(average access time)을 더 줄일 수 있는 여지가 생긴다는 것을 알 수 있었다. 따라서 본 논문에서 제시한 알고리즘은 다중 채널에서 더 빠른 채널에 접근 확률이 높은 데이타를 할당한다고 해도, 한 채널에 한 데이타가 오직 한 번씩만 할당된다면, 채널 내에서 그 데이타들의 접근 확률의 차이를 반영할 수 없었던 기존 알고리즘들의 한계를 보안한 것이다.

다중 채널 환경에서도 생길 수 있는 동일 채널 내에서 데이타의 접근확률의 차이를 반영하기 위하여, 본 논문에서는 정의한 비용 함수를 최소로 하는 할당 횟수를 찾는 데에 초점을 맞추었다. 각 분할이 정해질 때마다 모든 데이타에게는 그 분할 안에서, 즉 해당 채널 내에서 할당되어야 하는 횟수가 주어진다. 이와 동시에 채널 내에서 데이타의 할당 횟수에 따라 3.3장에서 설명한 BA방법으로 할당이 이루어진다. 따라서 다중 채널에서도 채널 각각이 다시 다중 디스크와 같은 형태를 띠게 되는 것이다. 이를 통해 실험 결과에서도 보듯이 기존 알고리즘에 비하여 평균 접근 시간을 상당히 줄일 수 있었다. 특히 채널수가 4이하로 작을 때는 기존 알고리즘들과 비교하여 평균 접근 시간이 더욱 줄어든다는 것을 알 수 있었다.

하지만 전체 주기(major cycle) 안에서의 할당 횟수를 얻기 위해서 본 논문에서 제시한 방법은 $O(N^2)$이상의 시간을 요구했다. 시스템에서 데이타의 잦은 갱신이 일어난다면, 감수해야할 이러한 수행시간 때문에 상당한 비용이 들 것이다. 따라서 다중 채널 환경에서도 채널 내의 접근 확률의 차이를 반영하되, 수행 시간을 더욱 줄이는 것이 향후 연구 과제로 남는다.

참 고 문 헌

[1] T. Imielinski and B. Badrinath, *Wireless Mobile Computing : Challenges in Data Management*, Comm. of ACM, Vol. 37, No. 10, pp. 18-28, 1994.

[2] K.L Tan and B.C. Ooi. *Batch Scheduling for Demand-driven Servers in Wireless Environment. Journal of Information Sciences, Elsevier Science Publishing Inc, North-Holland, Vol. 109, 281-298, 1998.*

[3] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik, *Broadcast Disks: Data Management for Asymmetric Communication Environments*, pp. 199-210 Proc. SIGMOD, May 1995.

[4] Wai Gen Yee, Student Member, IEEE, Shamkant B. Navathe, Edward Omiecinski, and Christopher Jermaine, *Efficient Data Allocation over Mulitple Channels at Broadcast Servers*, IEEE Transaction on computers, Vol. 51, No. 10, pp. 1231-1236 October 2002.

[5] K. Prabhakara, K.A. Hua, and J. Oh, *Multi-Level Multi-Channel Air Cache Designs for Broadcasting in a Mobile Environment*, Proc. Int'l Conf. Data Eng. pp. 167-176 (ICDE), 2000.

[6] W.C. Peng and M.S. Chen, *Dynamic Generation of Data Broadcasting Programs for Broadcast Disk Arrays in a Mobile Computing Environment*, Proc. ACM Conf. Information and Knowledge Management (CIKM), pp. 35-45, Nov. 2000.

[7] Dimitrios Katsaros, Yannis Manolopoulos, *Broadcast program generation for Webcasting*, Data & Knowledge Engineering 2003., 49(1), 1-21.

[8] Steven C. Chanpra, Raymond P. Canale, *Numerical Methods for Engineers with Programming and Software Applications*, McGraw-Hill, pp.153-159.

[9] W.G. Yee, E. Omiecinski, and S.B. Navathe, *Efficient Data Allocation for Broadcast Disk Arrays*, Technical Reprot GIT-CC-02-20, Georgia Inst. of Technology, 2001.

[10] D. Johnson, A. Demers, J.Ullman, M. Garey, and M. Graham. *Performance bounds for simple one-dimensional bin packing algorithms*. SIAM Jr. on Comput. 3(4):299-325, 1974.

[11] Chih-Hao Hsu, Guanling Lee and Arbee L.P. Chen, *Index and Data Allocation on Multiple Broadcast Channels Considering Data Access Frequencies*, Mobile Data Management, 2002. Proceedings of the Third International Conference on 8-11 Jan.2002 pp. 87-93.

[12] S. Acharya, M. Franklin, and S.Zdonik. *Balancing push and pull for data broadcast*. In Proc. of 1997 ACM SIGMOD, page 183-194, May 1997.

[13] N.H. Vaidya and S. Hameed, *Scheduling Data Broadcast in Asymmetric Communcation Environments*, J. Mobile Networks and Applications, vol. 5, pp. 171-182, 1999.

[14] W.C. Lee, Q. Hi, and D.L. Lee, *A Study on Channel Allocation for Data Dissemination in Mobile Computing Environments*, J. Mobile Networks and Applications, vol. 4, pp. 117-129, 1999.

[15] S. Hameed and N. Vaidya, *Efficient Algorithms for Scheduling Data Broadcastation. ACM/Baltzer Wireless Networks*, 5(3):183-293, 1999.

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 부교수. 관심분야는 Mobile Computing Systems, Mobile Databases, Telematics, Spatial DB, Mobile Agents, Streaming Data Processing in Ubiquitous Computing Environments, Distributed Databases

남 승 훈

1996년 3월~2003년 2월 서강대학교 수학과 학사. 2003년 3월~2005년 2월 서강대학교 컴퓨터학과 석사. 2005년 3월~현재 삼성전자 정보통신 총괄 무선 사업부 연구원. 관심분야는 이동통신, 모바일 컴퓨팅, 모바일데이타베이스, 모바일 트랜잭션

정 호 련

1999년 3월~2003년 2월 경희대학교 컴퓨터공학과 학사. 2004년 3월~현재 서강대학교 컴퓨터학과 석사과정. 관심분야는 이동통신, 모바일 컴퓨팅 시스템, 모바일 데이타베이스, 모바일 트랜잭션, 모바일 에이전트

이 원 택

1997년 3월~2004년 2월 서강대학교 컴퓨터학과 학사. 2005년 3월~현재 서강대학교 컴퓨터학과 석사과정. 관심분야는 이동통신, 모바일 컴퓨팅 시스템, 모바일 데이타베이스, 모바일 트랜잭션