A Binary-Number-Based Approach to Data Broadcasting in Wireless Information Systems*

Ye-In Chang, Che-Nan Yang and Jun-Hong Shen
Dept. of Computer Science and Engineering
National Sun Yat-Sen University
Kaohsiung, Taiwan, R.O.C
E-mail: changyi@cse.nsysu.edu.tw

Abstract

Acharya et al.'s Broadcast Disks (BD) is a periodic dissemination architecture in the context of mobile systems. However, BD has the empty slot problem, resulting in the waste of bandwidth and the increase of access time. Therefore, in this paper, we propose an efficient broadcast program, the Binary-Number-Based (BNB) approach, in which no empty slot is wasted with a restriction on the broadcast frequency. Next, we propose a generalized BNB (GBNB) approach in which no empty slot is wasted and the frequency of each disk is decided by the given access probabilities of pages on the disk. From our experimental results, we show that both our BNB and GBNB have shorter mean access time than BD.

(*keywords*: broadcast disks, broadcast schedule, mobile information systems)

1. Introduction

In a wireless mobile network, servers may have a relatively high bandwidth broadcast capability while clients cannot transmit or can do so only over a lower bandwidth (*e.g.*, cellular) link. Such systems have been proposed for many application domains, including traffic directions, news and stock quotes. In such *asymmetric communication* environments, a *push-based* architecture that exploits the relative abundance of downstream communication capacity (from the server to the clients) is proposed [1].

In a push-based information system, servers gather data items used frequently by mobile clients and broadcast the desired data items in the broadcast channel continuously and repeatedly [2, 4, 7]. The main advantage of broadcast delivery is its scalability: it is independent of the number of users the system is serving. In [2], Ammar and Wong, using a stochastic Markov Decision Process (MDP) formulation, concluded that the optimal schedule for a push-based broadcast will be periodic.

*This research was supported in part by the National Science Council of Republic of China under Grant No. NSC93-2213-E-110-003, and National Sun Yat-Sen University.

There have been many strategies proposed for efficient broadcast delivery [1, 5, 8]. Also, there has been some research on reducing access time by caching [1], where *access* time is the amount of time a client has to wait for a data item that it needs. Some other related work can be found in [6] for focusing on multiple wireless channels.

Among those strategies for efficient broadcast delivery, Acharya *et al.*'s Broadcast Disks [1] (BD) is one of well-known periodic *static* algorithms, where *static broadcast* means that a broadcast where the schedule of programs is fixed and even though the contents of a program can change with time. It constructs a broadcast program which can emphasize the most popular items and de-emphasize the less popular ones. For example, for the requests of data items A, B, and C, it could broadcast them as $ABAC$, $ABAC$, ..., given that data item A has a high priority (in terms of the access frequency) than data items B and C. The property of periodicity in a BD broadcast has several other advantages over a random broadcast program [1], including allowing to prefetch data, the use of "sleeping" to reduce power consumption [3], and providing periodicity. Periodicity may be important for providing correct semantics for updates and for introducing changes to the structure of the broadcast program. However, on the basis of BD, some broadcast slots may be unused, resulting in the waste of bandwidth and the increase of access time.

Therefore, in this paper, we propose the *Binary-Number-Based* (BNB) approach, in which no empty slot is wasted, and the broadcast frequency must be restricted to a value of 2^n, $n \geq 0$. Moreover, similar to the BD algorithm, the proposed BNB approach can guarantee the *equal space* property, where the *spacing* is the distance between two instances of an item. The argument in favor of a rigid enforcement of equal spacing is that for optimal periodic broadcast scheduling, all instances of an item should be equally *spaced* [9]. Next, we propose a generalized BNB (GBNB) approach in which no empty slot is wasted and the broadcast frequency of each disk is decided by the given demand access probabilities of pages on the disk. From

0-7803-9305-8/05/$20.00 ©2005 IEEE

our experimental results, we show that both of our BNB and GBNB approaches generate a small number of slots in one broadcast cycle and shorter mean access time than the BD algorithm does.

The rest of paper is organized as follows. In Section 2, we show an example of the empty slot problem in the BD algorithm. In Section 3, we present our BNB and GBNB approaches to solving the empty slot problem. The experimental results are presented in Section 4. Finally, we conclude this paper in Section 5.

2. An Example of the Empty-Slot Problem

Acharya *et al.* have proposed the use of a periodic dissemination architecture in the context of mobile systems. They call the architecture *Broadcast Disks* [1]. The algorithm has the following steps:

1. Order the pages (= D) from hottest (most popular) to coldest.
2. Partition the list of pages (= D) into multiple disks (= S disks), where each range (disk) i contains pages (= K_i) with similar access probabilities. That is, $D = \sum_{i=1}^{S} K_i$.
3. Choose the relative frequency R_i of broadcast for each disk i, $1 \leq i \leq S$.
4. Split each disk into a number of smaller units, called *chunks* C_{ij}, where C_{ij} denotes the j'th chunk in disk i. First, calculate L as the LCM (Least Common Multiple) of the relative frequencies. Then, split each disk i into $NC_i = L/R_i$ chunks, $1 \leq i \leq S$, where NC_i denotes the number of chunks in disk i.
5. Create the broadcast program by interleaving the chunks of each disk in the following manner:
   01 for $i := 1$ to L
   02 for $j := 1$ to S
   03 begin
   04 $k := ((i-1)\ mod\ NC_j) + 1$;
   05 Broadcast chunk $C_{j,k}$;
   06 end;

Figure 1 shows an example of the broadcast program generation, where $S = 3$, $D = 13$, $K_1 = 1$, $K_2 = 3$, $K_3 = 9$, $R_1 = 3$, $R_2 = 2$, and $R_3 = 1$. These disks are split into chunks according to Step 4 of the algorithm. That is, L (= $LCM(3,2,1)$) is 6, and we have $NC_1 = 2$, $NC_2 = 3$, $NC_3 = 6$. The resulting broadcast consists of 6 *minor cycles* (containing one chunk from each disk) which is the LCM of the relative frequencies. The resulting broadcast has a period of 24 pages with 6 empty slots.

3. The BNB Approach

On the basis of the BD algorithm, if the resulting assignment can satisfy the condition, $K_i\ mod\ NC_i = 0$, then there is no empty slot; that is, when $K_i/NC_i = K_i/(L/R_i) = (K_i \times R_i)/L$ results in no remainder, there is no empty slot. (In other words, when $(K_i \times R_i)\ mod\ L = 0$, there is no empty slot.) On the basis of this observation, first, we propose the Binary-Number-Based (BNB) approach to avoiding the empty slot problem, with a restriction on the value of the chosen relative frequency of each disk. Next, we propose a generalized BNB (GBNB) approach in which the broadcast frequency of each disk is decided by the given access frequency of each page on the disk. To make our work feasible, we have made some assumptions in our approach, including: (1) Each data item is in one page of the same size. (2) The client population and their access patterns do not change. (3) Data is read-only. (4) Clients make no use of their upstream communications capability. (5) When a client switches to the public channel, it can retrieve data pages immediately. (6) A query result contains only one page. (7) The server broadcasts pages over a single channel.

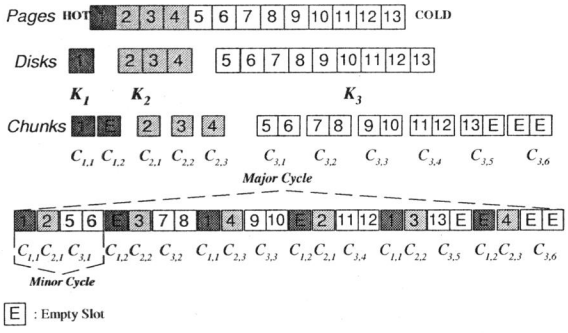

Figure 1: A broadcast program with 6 empty slots

3.1. The BNB Algorithm

Now, we present the proposed algorithm which partitions D pages into S broadcast disks such that no empty slot occurs, where $D \geq (2^S - 1)$. (Note that the limitation of D which must be greater than or equal to $(2^S - 1)$ is to ensure that we have enough pages to be assigned to S disks, which is proved in [10].) In the proposed algorithm, one new variable is used:

NS_i : the number of slots in a chunk of disk i, $1 \leq i \leq S$, *i.e.*,
$NS_i = \lceil \frac{K_i}{NC_i} \rceil = \lceil \frac{K_i}{L/R_i} \rceil = \lceil \frac{K_i \times R_i}{L} \rceil$;

The proposed algorithm is processed as follows:

1. Let $R_1 = 2^{S-1}$, $R_2 = 2^{S-2}$, ..., $R_S = 2^0$; that is, $R_i = 2^{S-i}$, $1 \leq i \leq S$. Therefore, we have $L = 2^{S-1}$.
2. Let D be represented in binary digits; that is, $D = (B_q B_{q-1} ... B_0)_2$, $B_i = 0$ or 1, $0 \leq i \leq q$ and $q = \lfloor \log_2 D \rfloor$. Let H be the number of 1's appearing in B_i, $0 \leq i \leq q$, and $1 \leq H \leq (q+1)$.
3. Let A_i denote the position of the i'th 1 appearing in $(B_q B_{q-1}\ ...\ B_0)_2$ from the right to the left, $1 \leq i \leq H$, $0 \leq A_i \leq q$.
4. Call *Procedure Assign* to partition the number of pages D to K_i, $1 \leq i \leq S$.
5. Split each K_i into a number of smaller units, called *chunks* denoted by C_{ij}, $1 \leq j \leq NC_i$, where NC_i (= L/R_i) is denoted by the number of chunks in disk i.

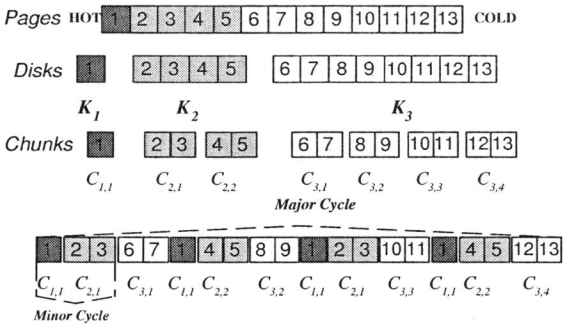

Figure 2: A broadcast program without empty slots

```
Procedure Assign;
begin
  if (H = S) then
    for i := 1 to S do K_i := 2^{A_i}
  else if (H < S) then Split
  else if (H > S) then Combine;
end;
```

Figure 3: The *Assign* procedure

6. Create the broadcast program by interleaving the chunks of each disk following the same way as Step 5 in Acharya *et al.*'s BD algorithm.

For the above algorithm, in the first step, we let $R_i = 2^{S-i}$, $1 \leq i \leq S$. In this way, $L = 2^{S-1}$. Then, to make $(K_i \times R_i)$ *mod* $2^{S-1} = 0$, the following conditions must be satisfied: (1) $K_i \geq (2^{S-1} \ / \ R_i) = 2^{S-1}/2^{S-i} = 2^{i-1}$ (*i.e.*, $K_i \geq 2^{i-1}$), and (2) (K_i *mod* $2^{i-1}) = 0$. In the second and third steps, we will make $D = \sum_{i=1}^{S} K_i$, where K_i equals the sum of some 2^W, $0 \leq W \leq q$. Therefore, we let D be represented in binary digits; that is why our proposed strategy is called the binary-number-based approach. Basically, we first aim to partition those D pages to H groups, with 2^W pages each group, and then, we try to assign those H groups of data items into S disks. We may need to apply the *split* or *combine* operations depending on $H < S$ or $H > S$, respectively. For example, in Figure 2, we have $S = 3$, $D = 13 = (1101)_2$ (which is the same input as the case shown in Figure 1) and let $R_1 = 2^{3-1} = 4$, $R_2 = 2^{3-2} = 2$, $R_3 = 2^{3-3} = 1$. Therefore, we have $L = 4$, $H = 3$, $A_1 = 0$, $A_2 = 2$ and $A_3 = 3$. For this example, if we assign $2^{A_1} = 2^0 = 1$ page to disk 1, $2^{A_2} = 2^2 = 4$ pages to disk 2, and $2^{A_3} = 2^3 = 8$ pages to disk 3, there is no empty slot in the resulting broadcast program, which is the case of $H = S$ and is the best case. Moreover, the broadcast cycle shown in Figure 2 contains 20 slots, shorter than 24 slots generated by the BD algorithm as shown in Figure 1. However, when $H \neq S$, we must carefully assign proper number of pages to each disk. We use Procedure *Assign*, as shown in Figure 3, to achieve this goal.

In Procedure *Assign*, three cases must be considered: (1) $H = S$: we let the first 2^{A_1} pages be assigned to disk 1, and then the next 2^{A_2} data items be assigned to disk 2, and so on; (2) $H < S$: we call Procedure *Split* to split 2^x pages into two 2^{x-1} pages and update $H := H + 1$, repeatedly, until $H = S$, where $1 \leq x \leq q$; (3) $H > S$: we call Procedure *Combine* to combine $2^x + 2^y$ pages into one group and update $H := H - 1$, repeatedly, until $H = S$, where $0 \leq x, y \leq (q-1)$. Because of the page limitation, the detail of Procedures *Split* and *Combine* is referred to [10]. The correctness (which guarantees no empty slots) of the proposed BNB approach is proved in [10].

3.2. The Generalized BNB (GBNB) Approach

As stated in [1], the only constraint on the relative broadcast frequencies of the disks is that they can be expressed as positive integers. Thus, it is possible to have arbitrarily fine distinctions in broadcasts such as a disk that rotates 141 times for every 98 times a slower disk rotates. However, this ratio results in a broadcast that has a very long period due to the result of LCM(141, 98). In addition, it is unlikely that such fine tuning will produce any significant performance benefit (*i.e.*, compared with a 3 to 2 ratio). Therefore, in practice, relative frequencies should be chosen with care and when possible, approximated to simpler ratios.

On the basis of the above comments, we can generalize the BNB approach such that the broadcast frequency of each disk is decided by the given demand access probabilities of pages on the disk, by modifying the step in deciding R_i. For each disk i, we let the mean of the demand access probabilities of pages in disk i be denoted by P_i, and $P_i = (\sum_{j=1}^{K_i} p_j / K_i)$. We then let $TempR_i = \sqrt{P_i}/\sqrt{P_S}$. That is, we will let $TempR_S = R_S = 1$. (Note that based on the lemma stated in [9], the minimum mean access time of the periodic broadcast is achieved when the page broadcast frequency $f_i \propto \sqrt{p_i}$, assuming that instances of each item are equally spaced. Therefore, to achieve the near optimal performance, we let $R_i \propto \sqrt{P_i}$.) Finally, we let R_i be a value of the power of 2 near $TempR_i$.

For example, given $TempR_1 = 7$, $TempR_2 = 2$ and $TempR_3 = 1$, we have $R_1 = 8$, $R_2 = 2$, and $R_3 = 1$. The correctness (which guarantees no empty slots) of the GBNB approach is proved in **Theorem 1**.

Theorem 1. *Under the condition $K_i \geq NC_i$, the GBNB approach which has $R_i = 2^j$ and $R_S = 1$, $1 \leq i \leq S, 0 \leq j \leq \log_2 K_S$, will guarantee no empty slot in the broadcast cycle [10].*

4. Performance

In this section, we study the performance of both of our BNB and GBNB approaches and make a comparison with Acharya *et al.*'s BD algorithm [1].

4.1. The Performance Model

When we simulate the process of the BD algorithm, the value of R_i's can be dependent on Δ, the broadcast shape parameter. Using Δ, the broadcast frequency R_i of each disk i can be computed relative to R_S, the broadcast frequency of the slowest disk (disk S) as follows [1]:

$$\frac{R_i}{R_S} = (S-i)\Delta + 1, \text{ and } R_S = 1,\ 1 \le i \le S.$$

When we simulate the process of the BD algorithm, the access probability, p_i of page i can be decided based on the *Zipf* distribution [1], which is typically used to model nonuniform access patterns. The *Zipf* distribution (for a specific D) can be expressed as

$$p_i = \frac{(1/i)^{\phi}}{\sum_{j=1}^{D}(1/j)^{\phi}},\ 1 \le i \le D, \quad (1)$$

where ϕ is a parameter named access skew coefficient or ***Zipf*** factor and $D \in N$. When we consider the demand access probability (from clients), we also apply the *Zipf* distribution with a *Zipf* factor γ. Here, we partition the pages into disks of K_i pages each, where $1 \le i \le S$, and we assume that the probability of accessing any page within a region is uniform; that is, the ***Zipf*** distribution is applied to these disks [1]. Therefore, we model the demand access probability of the ith disk (P_i) using the *Zipf* distribution as follows:

$$P_i = \frac{(1/i)^{\gamma}}{\sum_{j=1}^{S}(1/j)^{\gamma}}.$$

In this case, the first disk (K_1), which has the least number of records, is the most frequently accessed. Since each page i in disk k has the same demand access probability p_i, we have $p_i = P_k$, $1 \le k \le S$.

Two performance measures are considered in this comparison: (1) The total number of slots in one broadcast cycle. (2) The mean access time (or the expected time delay) which equals multiply the probability of access for each page i (p_i) with the expected delay for that page (EDP_i) and sum the results, where EDP_i denotes the average expected delay time for page i in disk k with the relative frequency $= R_k$.

Since when D is fixed, different values of K_i ($\sum_{i=1}^{S} K_i = D$) and R_i will result in different performance in Acharya *et al.*'s BD algorithm, in this comparison, we consider two possible cases in Acharya *et al.*'s BD algorithm:

1. **Fixed_K_i**: In this case, the value of K_i in the BD algorithm is the same as that in our BNB approach. While R_i in the BD algorithm is computed as follows [1]:

$$\frac{R_i}{R_S} = (S-i)\Delta + 1, \text{ and } R_S = 1,\ 1 \le i \le S.$$

2. **Fixed_R_i**: In this case, the value of R_i in the BD algorithm is the same as that in our BNB approach; that is, $R_i = 2^{S-i}$, $1 \le i \le S$. The value of K_i in the BD algorithm is computed by the *Zipf-like* distribution as follows [1]:

$$K_i = D \times \frac{(\frac{1}{S-i+1})^{\theta}}{\sum_{j=1}^{S}(1/j)^{\theta}}, \quad (2)$$

where θ is the *Zipf* factor. Here, K_1 has the fewest pages.

All the experimental results are the average of 1000 executions.

4.2. Performance Analysis

For the total number of slots (denoted by TS) in one major cycle in the BD algorithm, it can be computed as follows:

$TS = L \times \sum_{i=1}^{S} NS_i = L \times \sum_{i=1}^{S} \lceil \frac{K_i \times R_i}{L} \rceil$.

While the total number of slots in one major cycle in our BNB approach is computed as follows, where $L = 2^{S-1}$ and $R_i = 2^{S-i}$:

$TS = \sum_{i=1}^{S} K_i \times 2^{S-i}$.

The mean access time (denoted by $AccessT$) is calculated by multiplying the demand access probability for each page i (p_i) with the expected delay for that page (EDP_i) and summing the results. That is, $AccessT = \sum_{i=1}^{D} EDP_i \times p_i$.

Let SP_i denote the distance (*i.e.*, the number of slots) between the same page i in disk k occurring in a major cycle, where $SP_i = TS/R_k$ in both our BNB approach and the BD algorithm. For the mean access time (EDP_i) for page i in disk k in the BD algorithm and our BNB approach, it can be computed as follows:

$EDP_i = SP_i/2 = TS/(2 \times R_k)$.

In [9], this paper has proved that when instances of each item i are equally spaced with SP_i, and $SP_i = (\sum_{j=1}^{D} \sqrt{p_j \times l_j}) \times \sqrt{l_i/p_i}$, the minimum overall mean access time ($T_{optimal}$) is achieved and $T_{optimal} = 1/2(\sum_{i=1}^{D} \sqrt{p_i \times l_i})^2$. Note that we have $l_i(l_j) = 1$ based on our assumption, where $l_i(l_j)$ is the length of page i (j). We prove that SP_i derived from our BNB approach satisfies the above condition in [10]. Obviously, when $R_j = 1$, $EDP_i = TS/2$. Moreover, for any page i in disk j, it has the same EDP_i, since those pages in disk j have the same R_j and SP_i. Therefore, we let EDD_j denote the expected delay for any page i in disk j of K_j pages, where $EDD_j = \sum_{i=1}^{K_j} EDP_i$, $1 \le j \le S$. Therefore, the mean access time ($AccessT$) in the BD algorithm and our BNB approach is computed as follows:

$AccessT = \sum_{i=1}^{D} EDP_i \times p_i = \sum_{j=1}^{S} EDD_j \times K_j \times P_j$.

4.3. Simulation Results: BNB *vs.* BD

In this simulation, we let $\theta = 0.8$, $\gamma = 0.9$. We consider 6 test samples which include the combinations of $S = 2$, 3, and 4 and $\Delta = 3$, and 4, respectively, for a fixed D. For each test sample, we compute the average result for 1000 values of D which is a random value between 4000 and 5000.

In this simulation, we first consider the case of fixed_K_i with $R_S = 1$ in the BD algorithm. When $\Delta = 3$ and 4, the detailed simulation results about the total number of slots in one major cycle in the BNB approach and the BD algorithm are shown in Table 1, where BD denotes the BD algorithm, TWS

Table 1: A comparison of the total number of slots

S	BNB	BD		TWS *of* BD		$Max.\ TWS$	
Δ		3	4	3	4	3	4
2	4766	5286	5548	0.0	2.5	0	4
3	5034	6006	6523	40.7	71.0	65	120
4	5482	7188	9266	292.6	1574.0	484	2288

Table 2: A comparison of the mean access time (Fixed K_i)

S	BNB	$\Delta = 3$	$\Delta = 4$
2	828	884	918
3	476	539	577
4	335	411	523

Δ = (3, 4): the mean access time in the BD algorithm.

of BD denotes the total number of wasted slots in the BD algorithm, and $Max.\ TWS$ denotes the maximum number of wasted slots in the BD algorithm. From the results, we show that our BNB approach always generates a smaller number slots than the BD algorithm does. As Δ is increased, the total number of slots is increased in the BD algorithm. As S is increased, the total number of slots and the percentage of the total number of wasted slots are also increased in the BD algorithm. When $S = 4$ and $\Delta = 4$, up to 2288 slots are empty among 9266 slots in the BD algorithm.

A comparison of the average access time in the BNB approach and the BD algorithm is shown in Table 2. From this result, we show that the mean access time in our BNB approach is always shorter than that in the BD algorithm. As S is increased, the access time is decreased in both algorithm. As Δ is increased, the access time is increased in the BD algorithm.

Next, let's consider the case of fixed_R_i. A comparison of the average number of the total number of slots in the BNB approach and the BD algorithm is shown in Table 3. From this result, we show that our BNB approach always generates a smaller number slots than the BD algorithm does. As S is increased, the total number of slots and the percentage of the total number of wasted slots are increased in the BD algorithm. When $S = 4$, up to 17 slots are empty among 12530 slots in the BD algorithm.

A comparison of the mean access time in the BNB approach and the BD algorithm is shown in Figure 4. From this result, we show that the mean access time in our BNB approach is always shorter than that in the BD algorithm. As S is decreased, the mean access time is decreased in both algorithms.

Table 3: A comparison of the total number of slots (Fixed R_i)

S	BNB	BD	TWS *of* BD	$Max.\ TWS$
2	4766	6150	0.5	1
3	5034	8628	2.4	5
4	5482	12530	8.3	17

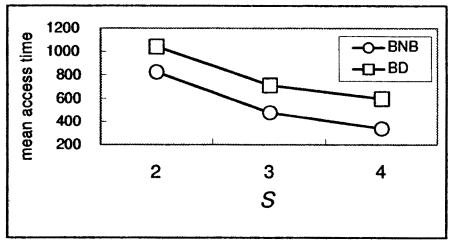

Figure 4: A comparison of the mean access time (Fixed R_i)

Table 4: A comparison of the mean access time between BNB and the optimal case

S	BNB	Opt
2	2175.802	2175.802
3	2050.627	2050.627
4	1819.691	1819.691

4.4. Simulation Results: BNB *vs.* the Optimal Case

Theorem 2. *If the demand access probabilities of pages are the same in each disk, the mean access time of the BNB approach is optimal.*

Proof. Because of the page limitation, the detail of this proof is referred to [10]. □

To calculate out the optimal value of the mean access time, *i.e.*, $T_{optimal} = 1/2(\sum_{i=1}^{D} \sqrt{p_i \times l_i})^2$ [9], we have to compute p_i first, $1 \le i \le D$. When the demand access probabilities of pages are the same in each disk, in our BNB approach, we have $p_i = P_j$, $1 \le j \le S$ and $1 \le i \le K_j$. Moreover, based on [9], the minimum mean access time is achieved when the disk broadcast frequency $R_i \propto \sqrt{P_i}$, assuming that instances of each item are equally spaced. That is, $R_i^2 \propto P_i$. Therefore, in our BNB approach,

$$\frac{P_i}{P_{i+1}} = \frac{R_i^2}{R_{i+1}^2} = (\frac{2^{S-i}}{2^{S-i-1}})^2 = 2^2 = 4,\ 1 \le i < S.$$

It also means that $P_i = 4 \times P_{i+1}$. We assume that the sum of demand access probabilities of pages in the broadcast file is equal to 1; therefore, we have $\sum_{i=1}^{S} K_i P_i = 1$. Furthermore, in our BNB approach, the total number of pages (K_i) in each disk i are known. The mean access time of the BNB approach (under some conditions) and the optimal mean access time are shown in Table 4, where *Opt* represents the optimal case. It shows that the mean access time of the BNB approach is optimal under the condition that the demand access probabilities of pages are the same in each disk.

4.5. Simulation Results: GBNB, BD and the Optimal Case

In the BD algorithm, the total number of pages (K_i) in each disk i follows a *Zipf* distribution according to equation (2), while K_i in our GBNB approach is computed from procedure *Assign*. Next,

Table 5: A comparison of the total number of slots between GBNB and BD

S	GBNB	BD	TWS of BD	$Max.\ TWS$
2	5283.04	7793.33	0.952	2
3	5479.47	7328.43	2.934	6
4	5706.90	7889.06	15.827	29

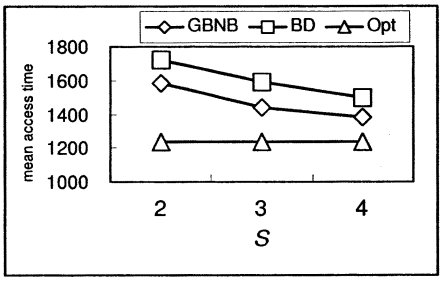

Figure 5: A comparison of the mean access time among GBNB, BD and Opt

for each strategy, based on the related K_i (which is different in these two strategies), we calculate out P_i $(= \sum_{j=1}^{K_i} p_j/K_i)$. On the basis of [9], the minimum mean access time is achieved when the broadcast frequency of the page is proportional to $\sqrt{p_i}$. Therefore, we assign the broadcast frequency of each disk i with $\sqrt{P_i}$.

In the BD algorithm, we let $R_S = 1$ and $R_i = \sqrt{P_i}/\sqrt{P_S}$, $1 \leq i \leq S$. Finally, we round the relative frequency (R_i) of each disk i to an integer. While in our GBNB approach, we will adjust R_i to be a value of the power of 2. A comparison of the total number of slots between the GBNB approach and the BD algorithm is listed in Table 5. The comparison of the mean access time among our GBNB, BD and the optimal case is shown in Figure 5, where the average number of pages = 4505 and $\phi = 0.9$. From Figure 5, we observe that the mean access time of the GBNB approach is shorter than that of the BD algorithm. The main reason is that our GBNB approach guarantees no empty slots. Since our GBNB only guarantees no empty slots and *equal space*, but the spacing SP_i does not satisfy equation

$$SP_i = (\sum_{j=1}^{D} \sqrt{p_j}) \frac{1}{\sqrt{p_i}}$$

in [9], the mean access time of GBNB, of course, is longer than that of the optimal case.

5. Conclusion

Acharya *et al.*'s BD algorithm has the empty slot problem, resulting in the waste of bandwidth and the increase of access time. In this paper, we have presented the Binary-Number-Based (BNB) approach, in which no empty slot is wasted. Although there is some restriction on the chosen relative broadcast frequency, the proposed approach guarantees the equal space property, which is helpful in some issues. Next, we have presented the GBNB approach in which the broadcast frequency of each disk is decided by the given demand access probabilities of pages on the disk. From our performance analysis and simulation, we have shown that both of our BNB and GBNB approaches generate a small number of slots in one major cycle and shorter mean access time than the BD algorithm.

References

[1] S. Acharya, M. Franklin, S. Zdonik and R. Alonso, "Broadcast Disks: Data Management for Asymmetric Communications Environments," *Proc. of ACM SIGMOD Int. Conf. Management of Data*, pp. 199–210, 1995.

[2] M. H. Ammar and J. W. Wong, "On the Optimality of Cyclic Transmission in Teletext Systems," *IEEE Trans. on Communications*, Vol. 35, No. 1, pp. 68–73, Jan. 1987.

[3] D. Barbara, "Mobile Computing and Database–A Survey," *IEEE Trans. on Knowledge and Data Eng.*, Vol. 11, No. 1, pp. 108–117, Jan./Feb. 1999.

[4] A. Bar-Noy, B. Patt-Shamir and I. Ziper, "Broadcast Disks with Polynomial Cost Functions," *Wireless Networks*, Vol. 10, No. 2, pp. 157–168, Mar. 2004.

[5] Y. I. Chang and C. N. Yang, "A Complementary Approach to Data Broadcasting in Mobile Information Systems," *Data and Knowledge Engineering*, Vol. 40, No. 2, pp. 181–194, Feb. 2002.

[6] Y. I. Chang and S. Y. Chiu, "A Hybrid Approach to Query Sets Broadcasting Scheduling for Multiple Channels in Mobile Information Systems," *Journal of Information Science and Engineering*, Vol. 18, No. 5, pp. 641–666, Sept. 2002.

[7] C. L. Hu and M. S. Chen, "Adaptive Information Dissemination: An Extended Wireless Data Broadcasting Scheme with Loan-Based Feedback Control," *IEEE Trans. on Mobile Computing*, Vol. 2, No. 4, pp. 322–336, Oct. 2003.

[8] W. C. Peng, J. L. Huang, and M. S. Chen, "Dynamic Leveling: Adaptive Data Broadcasting in a Mobile Computing Environment," *Mobile Networks and Applications*, Vol. 8, No. 4, pp. 355–364, Aug. 2003.

[9] N. H. Vaidya and S. Hameed, "Scheduling Data Broadcast in Asymmetric Communication Environments," *Wireless Networks*, Vol. 5, No. 3, pp. 171–182, May 1999.

[10] C. N. Yang, "A Compementary Approach to Data Broadcasting in Mobile Information Systems," *Master Thesis, Dept. of Computer Science and Engineering, National Sun Yat-Sen University*, June 2000.