An Adaptive Approach to Data Broadcasting in Mobile Information Systems[*]

Ye-In Chang, Shih-Ying Chiu 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

Broadcast data delivery rapidly becomes the choice for disseminating information to a massive user population. The main advantage of broadcast delivery is its scalability. Acharya et al.'s Broadcast Disks is one of well-known static algorithms for efficient broadcast delivery. However, based on Acharya et al.'s algorithm, some broadcast slots may be unused, which results in the waste of bandwidth and the increase of access time. Therefore, in this paper, we propose an efficient broadcast program, the adaptive approach, which prevents the occurrence of empty slots. The basic idea of the adaptive approach is to dynamically arrange the number of slots in each chunk on a disk so that it is impossible for empty slots to occur, where a chunk is the unit for broadcast from a disk per time. From the simulation results, our adaptive approach generates a smaller number of slots in a broadcast cycle and shorter mean access time than Acharya et al.'s algorithm.

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

1. Introduction

Mobile computing has become a reality due to the convergence of two technologies: the appearance of powerful portable computers and the development of fast reliable networks [5, 19]. In the mobile wireless computing environment, there are two sets of entities: servers and mobile clients (MCs) [11]. In general, servers are powerful stationary machines with database systems, and MCs are portable computing devices, *e.g.,* palmtop or laptop computers. There are some characteristic features of the mobile wireless computing system, making it different from the traditional wired system [5, 22]: (1) asymmetry in communications; (2) frequent disconnections; (3) power limitations; (4) screen size.

In the evolving field of mobile computing, there is a growing concern to provide mobile users with timely access to large amounts of information [12, 17]. Examples of such services include weather, high-way condition, traffic directions, news and stock quotes. Each MC will retrieve data pages via wireless information channels. The wireless channels consists of two distinct sets of channels: uplink channels (clients to servers) and downlink (servers to clients) channels. In the environment under consideration, the downstream communication capacity is relatively much greater than the upstream communication capacity. Such environments are, hence, called asymmetric communication environment [2].

In such asymmetric communication environments, using push-based systems is an efficient way to disseminate a large amount of information to MCs [20, 21, 25]. The push-based system repeatedly broadcasts data stream to unspecified clients via the downlink channel, and they access the data of interest in the broadcast stream [13]. In the push-based systems, the number of mobile clients which can simultaneously listen to the downlink channel can be almost scaled. Using of push-based systems not only can save the bandwidth of the uplink channel and but also can be scaled to any number of mobile clients who listen to the publishing report.

To disseminate data via broadcasting, a server constructs a broadcast program and periodically transmits data according to the program [8, 9, 18]. The motivation for that work was teletext systems [4]. In [4], Ammar and Wong, using a stochas-

[*]This research was supported in part by the National Science Council of Republic of China under Grant No. NSC-89-2218-E-110-004 and National Sun Yat-Sen University.

tic Markov Decision Process (MDP) formulation, concluded that the optimal schedule for a push-based broadcast will be periodic. In a uniform broadcast program, all the objects are broadcast once in a broadcast cycle regardless of their access frequencies, as was done in Datacycle [6]. As a result, the average access time (the time elapsed from the moment a client requests for an object to the point when the desired object is download by the client) of an object will, on average, be half the time between successive broadcasts of the data file. On the contrary, a nonuniform broadcast program favors objects with higher access frequencies. Hence, in a broadcast cycle of a nonuniform broadcast, while all objects are broadcast, some will appear more often than others. The resulting effect is that objects that are more frequently broadcast will have a shorter access time than those are less frequently broadcast [23].

Over the past few years, a considerable number of studies has been conducted on efficient delivery on the wireless broadcast. For the uniform broadcast in which the same data item appears once in a broadcast cycle, mobile users may query a set of dependent data items in a query, not only one data item. According to [16], prior research work of dependent data broadcast can be categorized by the following two properties: (1) the number of broadcast channels considered, single or multiple channels, and (2) the constraint of the retrieval sequence of data items in each query, ordered, or unordered queries. Several studies in [14, 20] have been made on the broadcast schedule for unordered queries on the single broadcast channel. In [10], they concentrated on the broadcast schedule for ordered queries on the single broadcast channel. In [8, 16], they have studied the issue on data broadcast for unordered queries in a multi-channel mobile environment. Research work in [15] considered data broadcast for ordered queries on the multiple channels. For nonuniform broadcast in which data items are broadcast according to the access frequency, Acharya *et al.* [1] proposed Broadcast Disks in a single-channel mobile environment. Chang and Yang [7] solved the empty slot problem in Acharya *et al.*'s Broadcast Disks. The study in [22] focused on partitioning data with skewed access patterns into multiple broadcast channels.

Among those strategies for efficient broadcast delivery, Acharya *et al.*'s Broadcast Disks [3] is one of well-known static algorithms. It constructs a broadcast program which can emphasize the most popular items and de-emphasize the less popular ones. In this way, one can establish a trade-off between *access time* for high-priority data and that of the low-priority items. However, based on Acharya *et al.*'s approach, some broadcast slots may be unused, which results in the waste of bandwidth and the increase of access time. Therefore, in this paper, we propose an efficient broadcast program, the *adaptive* approach, in which the case of empty slots is impossible to occur. Based on our approach, not only there is no empty slots to occur in a broadcast cycle, but also the total number of slots in a broadcast cycle is smaller than Acharya *et al.*'s algorithm. From our performance analysis and simulation, we show that our adaptive approach generates a smaller number of slots in one broadcast cycle and shorter mean access time than Acharya *et al.*'s algorithm.

The rest of paper is organized as follows. In section 2, we give a brief description of Acharya *et al.*'s algorithm and show an example of the empty slot problem in Acharya *et al.*'s algorithm. In section 3, we present our adaptive approach to avoid the occurrence of empty slots. In section 4, we study the performance of our adaptive approach, and make a comparison with Acharya *et al.*'s algorithm. Finally, section 5 gives the conclusion.

2. Background

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, 3]. It can construct a memory hierarchy in which the highest level contains a few items and broadcasts them with high frequency while subsequent levels contain more and more items and broadcast them with less and less frequency. The architecture imposes a Multi-disk structure on the broadcast medium in a way that allows substantial flexibility in fitting the relative broadcast frequencies of data items to the access probabilities of a client population. The broadcast is created by assigning data items to different "disks" of varying sizes and speeds, and then multiplexing the disks on the broadcast channel. Items stored on faster disks are broadcast more often than items on slower disks. Given the amount of data pages D and the number of multiple disks S, the algorithm has the following steps:

1. Order the pages ($= D$) from the hottest (the most popular) one to the coldest one.

2. Partition these ordered pages ($= D$) into mul-

tiple disks (= S disks), where each range contains pages with similar access probabilities.

3. Choose the relative frequency R_i of broadcast for each disk i. For example, given two disks, if we choose $R_1 = 3$, and $R_2 = 2$, it means that disk 1 could be broadcast three times for every two times that disk 2 is broadcast.

4. Partition each disk into a number of smaller units, called *chunks* C_{ij}, where C_{ij} denotes the jth 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, where NC_i denotes the number of chunks in disk i. In the previous example, NC_1 would be 2, while NC_2 would be 3.

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      $k := ((i-1) \bmod NC_j) + 1$;
05      Broadcast chunk $C_{j,k}$;
06   end;
```

Figure 1 shows an example of a broadcast program generated by Acharya *et al.*'s algorithm, in which several empty slots can occur. Given $D = 15$ and $S = 3$, 15 pages are sorted from the hottest one to the coldest one. Next, these 15 pages are partitioned into 3 groups according to their relative frequencies. Let K_i denote the number of pages on disk i: $K_1 = 2$, $K_2 = 4$, and $K_3 = 9$. Moreover, $R_1 = 3$, $R_2 = 2$, and $R_3 = 1$. Then, these disks are split into chunks according to Step 4 of the algorithm. That is, L is 6, so we have $NC_1 = 2$, $NC_2 = 3$, and $NC_3 = 6$. The resulting broadcast program consists of 6 *minor cycles*, and has a period of 30 slots with 7 empty slots. Note that the chunks of different disks can be of different sizes, but the size of chunks in a disk is fixed. Finally, the resulting broadcast consists of 6 *minor cycles* (containing one chunk from each disk) which is the LCM of the relative frequencies.

3. The Adaptive Approach

In the multi-disks architecture of Acharya *et al.*'s algorithm, each disk will spin out one unit of a chunk every time. The number of chunks on each disk can be different, but the number of slots in each chunk on a disk is the same. That is why empty slots may occur. In our approach, we can prevent the occurrence of empty slots as we dynamically arrange the number of slots in each chunk on a disk.

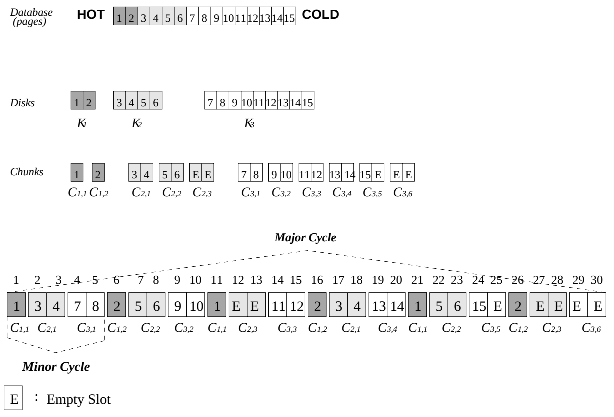

Figure 1: A broadcast program with 7 empty slots based on Acharya *et al.*'s algorithm

By observing Acharya *et al.*'s algorithm, we find two interesting phenomena. One is that empty slots occur when $K_i \bmod NC_i \neq 0$, where K_i is the number of pages in disk i and NC_i is the number of chunks in disk i, and the other is that the number of slots in every chunk on a disk is always the same. Let NSC_{ij} be the number of slots in the jth chunk of disk i. When $K_i \bmod NC_i \neq 0$, NSC_{ij} is always equal to $\lceil K_i/NC_i \rceil$. In this case, the total number of slots in disk i will be greater than the number of pages in disk i ($\sum_{j=1}^{NC_i} NSC_{ij} > K_i$) and empty slots can occur. The basic idea of the adaptive approach is to adjust NSC_{ij} such that $\sum_{j=1}^{NC_i} NSC_{ij} = K_i$. Therefore, finally, the total number of slots in a major cycle will be less than or equal to the one computed from Acharya *et al.*'s algorithm.

For the previous example shown in Figure 1, since we have $K_2 = 4$ and $NC_2 = 3$, resulting in $K_2 \bmod NC_2 \neq 0$, two empty slots occur in disk 2. If we let the number of slots in both chunks 2 and 3 on *disk* 2 be one, *i.e.,* C_{22} containing page 5 and C_{23} containing page 6, then there will be no empty slots on disk 2, where C_{ij} denotes the jth chunk in disk i. That is, we can prevent the occurrence of empty slots by adopting different number of slots in each chunk on a disk.

3.1. Assumptions

This paper focuses on wireless broadcast environment. Some assumptions should be restricted in order to make our work feasible [11]. These assumptions include:

1. The client population and their access patterns do not change. This implies that the content and the organization of the broadcast program remains static.
2. Data is read-only; there are no pages updated either by the clients or at the servers.
3. Clients make no use of their upstream communications capability and retrieve required data items from the broadcast; they provide feedback to servers and there is no prefetching.
4. Clients are simple and without a great amount of memory; there is no cache scheme on the clients.
5. When a client switches to the public channel, it can retrieve data pages immediately. The delay for hardware/software preparation to begin monitoring the broadcast channel is so short to be ignored.
6. The broadcast infrastructure is reliable; each item transmitted by the server is always received correctly by each clients.
7. A query result contains only one page.
8. The server broadcasts pages over a single channel. All clients retrieve data pages from this single channel.
9. The length of each page is fixed. This assumption makes the time slots of each page are equal.
10. The number of pages (K_i) in a disk i has to be greater than or equal to the number of chunks (NC_i) in a disk i. That is, the condition, $K_i > NC_i$, must be satisfied.

3.2. The Algorithm

Now, we present the proposed algorithm which partitions D pages into S broadcast disks such that no empty slot occurs. In the proposed algorithm, the following variables are used:

1. D: the number of pages;
2. P_i: the ith page in a decreasing order of demand frequency, $1 \leq i \leq D$;
3. S: the number of disks;
4. R_i: the relative frequency of disk i, $1 \leq i \leq S$;
5. L: the least common multiple of R_i, $1 \leq i \leq S$, *i.e.*, L = LCM (R_1, R_2, ..., R_S);
6. K_i: the number of pages in disk i, $1 \leq i \leq S$, and $\sum_{i=1}^{S} K_i = D$;
7. NC_i: the number of chunks in disk i, and $NC_i = L/R_i$, $1 \leq i \leq S$;
8. NSC_{ij}: the number of slots in the jth chunk in disk i, $1 \leq j \leq NC_i$, $1 \leq i \leq S$;
9. C_{ij}: the jth chunk in disk i, $1 \leq i \leq S$.
10. O_{ijk}: the kth slot of the jth chunk in disk i, $1 \leq i \leq S$.

The proposed algorithm is processed as follows:

1. Order the pages from the hottest (most popular) one to the coldest one.
2. Partition the list of pages into multiple disks (= S disks), provided with $K_i < K_{i+1}$, $1 \leq i < S$, where each range contains pages with similar access probabilities.
3. Choose the relative frequency R_i of broadcast for each disk i, provided with $R_i > R_{i+1}$, $1 \leq i < S$.
4. Partition each disk into a number of smaller units, called *chunks* C_{ij}, where C_{ij} denotes the jth 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, where NC_i denotes the number of chunks in disk i.
5. Call Procedure *Partition* to decide the number of slots in every C_{ij}, called NSC_{ij}, where NSC_{ij} denotes the number of slots of the jth chunk in disk i.
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.

From Step 1 to Step 4, the adaptive approach decides the value of K_i, R_i, L, and NC_i. In Step 5, we calculate NSC_{ij} by processing the *Partition* procedure as shown in Figure 2. Note that in Acharya *et al.*'s algorithm, for a certain disk i, NSC_{ij} always equals $\lceil K_i/NC_i \rceil$, $1 \leq j \leq NC_i$. In the *Partition* procedure, if (K_i mod NC_i) = 0, then we have $NSC_{ij} = \lceil K_i/NC_i \rceil = \lfloor K_i/NC_i \rfloor$, $1 \leq j \leq NC_i$. In this case , the result of NSC_{ij} is the same as that of the Acharya *et al.*'s algorithm. If (K_i mod NC_i) $\neq 0$, then we need to adjust NSC_{ij} to an appropriate value such that $\sum_{j=1}^{NC_i} NSC_{ij} = K_i$ and $NSC_{ij} \geq NSC_{i(j+1)}$, $1 \leq j < NC_i$. Basically, K_i can be divided into two groups in the following ways, if $\lfloor K_i/NC_i \rfloor = \lceil K_i/NC_i \rceil - 1$, *i.e.,* K_i mod $NC_i \neq 0$, where $\lceil K_i/NC_i \rceil = b$ and (K_i mod NC_i) = a:

$$\begin{aligned} K_i &= \lfloor K_i/NC_i \rfloor \times NC_i + (K_i \bmod NC_i) \\ &= (\lceil K_i/NC_i \rceil - 1) \times NC_i + (K_i \bmod NC_i) \\ &= (b-1) \times NC_i + a \\ &= (b-1) \times a + (b-1) \times (NC_i - a) + a \\ &= \textstyle\sum_{j=1}^{a}(b-1) + \sum_{j=a+1}^{NC_i}(b-1) + \sum_{j=1}^{a} 1 \\ &= \textstyle\sum_{j=1}^{a}((b-1)+1) + \sum_{j=a+1}^{NC_i}(b-1) \\ &= \textstyle\sum_{j=1}^{a} b + \sum_{j=a+1}^{NC_i}(b-1). \end{aligned}$$

Therefore, for a certain disk i, there are two possible values of NSC_{ij}, *i.e.,* $\lceil K_i/NC_i \rceil$ or $\lceil K_i/NC_i \rceil - 1$. That is, we can divide these NC_i chunks into two groups: one is from the first chunk to the ath chunk, and the other is from the $(a+1)$th chunk to the last chunk. The number of slots in each chunk in the first group equals b, while the number of slots in the other group equals $(b-1)$. Therefore, there will be no empty slots in each chunk.

For the same input data, $S = 3$ and $D = 15$, as shown in Figure 1, Figure 3 shows the result based on our adaptive approach, in which no empty slots occur. First, we order these 15 pages from the hottest one to the coldest one. Second, we partition these 15 pages into 3 disks and let $K_1 = 2$, $K_2 = 4$, and $K_3 = 9$. Third, given $R_1 = 3$, $R_2 = 2$, and $R_3 = 1$, we have $L = LCM(3, 2, 1) = 6$. Fourth, we figure out that $NC_1 = L/R_1 = 2$, $NC_2 = L/R_2 = 3$ and $NC_3 = L/R_3 = 6$. Fifth, we call the *Partition* procedure to decide the number of slots in every chunk on each disk, *i.e.,* to decide the value of NSC_{ij}. Take disk 3 as an example, we have $NC_3 = \lceil 6/1 \rceil = 6$ chunks. Therefore, we let $a = (K_3 \bmod NC_3) = 9 \bmod 6 = 3$ and partition these 6 chunks into two groups. The first group is from the first chunk to the third chunk, while the second group is from the fourth chunk to the sixth chunk. Therefore, for $j = 1$ to 3, we have $NSC_{3j} = b = \lceil K_3/NC_3 \rceil = 2$; in the meanwhile, for $j = 4$ to 6, we have $NSC_{3j} = b - 1 = \lceil K_3/NC_3 \rceil - 1 = 1$. Finally, the resulting broadcast program consists of 6 *minor cycles*, and has a period of 23 slots without any empty slot.

```
Procedure Partition;
begin
  for $i$ := 1 to $S$ do
  begin
    if ($K_i$ mod $NC_i = 0$) then
      for $j$ := 1 to $NC_i$ do
        $NSC_{ij}$ := $K_i$ div $NC_i$
    else
    begin
      $a := K_i$ mod $NC_i$;
      $b := \lceil \frac{K_i}{NC_i} \rceil$;
      for $j$ := 1 to $a$ do
        $NSC_{ij} := b$;
      for $j$ := $(a+1)$ to $NC_i$ do
        $NSC_{ij} := b - 1$;
    end;
  end;
end;
```

Figure 2: The *Partition* procedure

As compared with the result based on Acharya *et al.*'s algorithm shown in Figure 1, the size of each minor cycle in our adaptive approach is not the same and is always less than or equal to that in Acharya *et al.*'s algorithm, and no empty slots can occur. Obviously, as compared with Acharya *et al.*'s algorithm, the total number of slots in a broadcast cycle in our adaptive approach, which can be computed by summing the size of all minor cycles, is less than or the same as that in Acharya *et al.*'s algorithm. Therefore, the expected mean access time in our adaptive approach will also be shorter than or equal to that in Acharya *et al.*'s algorithm, which will also be verified by the performance study as discussed later.

4. Performance Study

In this section, we study the performance of our complementary approach and make a comparison with Acharya *et al.*'s algorithm. Our experiments were performed on a PentiumIII 733 MHz, 128 MB of main memory, running Windows ME.

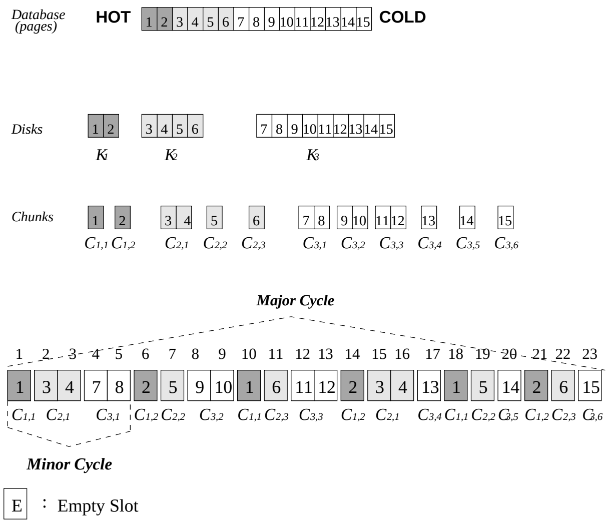

Figure 3: A broadcast program based on the adaptive approach

Table 1: Parameters used in the simulation

S	the number of disks
D	the number of distinct pages to be broadcast
K_i	the number of pages in disk i
R_i	the relative frequency of disk i
Δ	the broadcast shape parameter
θ	the *Zipf* factor for partition size
γ	the *Zipf* factor for frequency of access

4.1. The Simulation Model

The parameters used in the model are shown in Table 1. S is the number of disks in the broadcast program, and D is the number of distinct pages to be broadcast. K_i is the number of pages in disk i; therefore, the sum of K_i over all i is equal to D, where $1 \leq i \leq S$ (*i.e.*, $D = \sum_{j=1}^{S} K_i$). R_i is the relative frequency of disk i, where $1 \leq i \leq S$. When we simulate the process of Acharya *et al.*'s algorithm, we need to decide the values of R_i's, which can be dependent on Δ. That is, Δ is a factor used to measure the relative frequencies of broadcast of each disk. Using Δ, the frequency of broadcast R_i of each disk i, can be computed relative to R_S, the broadcast frequency of the slowest disk (disk S) as follows [3, 23]:

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

When Δ is zero, the broadcast is flat: all disks spin at the same speed. As Δ is increased, the speed differentials among the disks increase. For example, for a 3-disk broadcast, when $\Delta = 4$, the relative frequencies are 9, 5, and 1 for disks 1, 2, and 3, respectively. Table 2 shows examples of the relative frequencies and their LCM when $\Delta = 4$ and 5.

Table 2: Relative frequencies (( ): LCM)

S	$\Delta = 4$	$\Delta = 5$
3	1, 5, 9 (45)	1, 6, 11 (66)
4	1, 5, 9, 13 (585)	1, 6, 11, 16 (528)

Moreover, when we simulate the process of Acharya *et al.*'s algorithm, we need to decide the values of K_i's, which can be decided based on the *Zipf* distribution [3, 23]. The *Zipf* distribution is typically used to model nonuniform access patterns [3]. The *Zipf* distribution can be expressed as $p_i = \frac{(1/i)^\theta}{\sum_{j=1}^{M}(1/j)^\theta}$, $1 \leq i \leq M$, where θ is a parameter named access skew coefficient or *Zipf* factor and $M \in N$. Different values of θ yield different *Zipf* distribution. When $\theta = 0$, we have the uniform distribution. When $\theta = 1$, we have the highly nonuniform *Zipf* distribution. That is, it produces access patterns that become increasingly skewed as θ increases—the probability of accessing any page numbered i is proportional to $(1/i)^\theta$. For example, when $M = 3$, $\theta = 1$, we have $p_1 = \frac{6}{11}$, $p_2 = \frac{3}{11}$, and $p_3 = \frac{2}{11}$. Therefore, K_i in Acharya *et al.*'s algorithm can be decided based on the *Zipf-like* distribution as follows [3, 23]:

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

Here, K_1 has the fewest pages, K_2 has the next fewest pages, and K_S has the most number of pages.

When we consider the demand frequency of data access for page i (denoted by DFP_i), we also apply the *Zipf* distribution with a *Zipf* factor γ. Here, we partition the pages into regions (= number of disks) of K_i pages each, where $1 \leq i \leq S$, and we assume that the probability of accessing any page within a region is uniform; that is, the *Zipf* distribution is applied to these regions [3]. Therefore, we model the demand frequency of access of the ith disk (DFD_i) using the *Zipf* distribution as follows:

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

where γ is the *Zipf* factor of the *Zipf* distribution. In this case, the first disk (K_1), which has the least number of records, is the most frequently accessed, the second disk (K_2) is next, and so on. Since each page w in disk i has the same demand frequency DFP_w, we have $DFP_w = DFP_i$, $i \le i \le S$.

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 (DFP_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 $= R_k$.

4.2. Performance Analysis

For the total number of slots (denoted by TS) in one broadcast cycle, it can be computed by summing the total number of slots in each minor cycle (denoted by MS_m, the mth minor cycle). There are L minor cycles in a major cycle. Let the size of a minor cycle be denoted by MS_m, $1 \le m \le L$. Based on the Acharya *et al.*'s algorithm, the size of each minor cycle is the same, *i.e.,* $MS_m = \sum_{i=1}^{S} NS_i$, $1 \le m \le L$, where NS_i denotes the number of slots in a chunk of disk i, $1 \le i \le S$. Therefore,

$$\begin{aligned} TS &= \textstyle\sum_{m=1}^{L} MS_m \\ &= \textstyle\sum_{m=1}^{L} \sum_{i=1}^{S} NS_i \\ &= L \times \textstyle\sum_{i=1}^{S} NS_i. \end{aligned}$$

However, in the adaptive algorithm, the size of each minor cycle is different and can be computed as follows.

$$MS_m = \textstyle\sum_{p=1}^{S} NSC_{p,q},$$

where $q = ((m-1) \bmod NC_p) + 1$, $1 \le p \le S$, $1 \le m \le L$.

For instance, for the example shown in Figure 3, the number of slots in the first minor cycle is different from that in the second minor cycle. For the first minor cycle:

$$\begin{aligned} MS_1 &= \textstyle\sum_{p=1}^{3} NSC_{p,q} \\ &= NSC_{1,1} + NSC_{2,1} + NSC_{3,1} \\ &= 1 + 2 + 2 \\ &= 5. \end{aligned}$$

For the second minor cycle:

$$\begin{aligned} MS_2 &= \textstyle\sum_{p=1}^{3} NSC_{p,q} \\ &= NSC_{1,2} + NSC_{2,2} + NSC_{3,2} \\ &= 1 + 1 + 2 \\ &= 4. \end{aligned}$$

Similar to the Acharya *et al.*'s algorithm, TS in the adaptive algorithm can be calculated by summing the total number of slots in each minor cycle, that is:

$$\begin{aligned} TS &= \textstyle\sum_{m=1}^{L} MS_m \\ &= \textstyle\sum_{m=1}^{L} (\sum_{p=1}^{S} NSC_{p,q}), \end{aligned}$$

where $q = ((m-1) \bmod NC_p) + 1$. For instance, for the example shown in Figure 3, the total number of a broadcast cycle is:

$$\begin{aligned} TS &= \textstyle\sum_{m=1}^{6} MS_m \\ &= MS_1 + MS_2 + MS_3 + MS_4 + MS_5 + MS_6 \\ &= 5 + 4 + 4 + 4 + 3 + 3 \\ &= 23. \end{aligned}$$

We can find that in the adaptive approach, the size of each minor cycle is either equal to or less than that of Acharya *et al.*'s algorithm. Obviously, TS in the adaptive approach is always less than or equal to that in Acharya *et al.*'s algorithm.

The mean access time (denoted by $AccessT$) is calculated by multiplying the probability of access for each page a (DFP_a) with the expected delay for that page (EDP_a) and summing the results. That is, $AccessT = \sum_{a=1}^{D} EDP_a \times DFP_a$. For each page a, we let O_{ijk} to represent the page arranged in the kth slot of the jth chunk in the ith disk. For instance, for the example shown in Figure 3, page 6 is in the first slot of the third chunk in the second disk, so we use O_{231} to denote it.

Page O_{ijk} will occur R_i times in a major cycle and occur respectively in the $(j + \frac{LCM}{R_i} \times Z)'$th minor cycle, where $0 \le Z \le (R_i - 1)$. Let $M^m_{O_{ijk}}$ represent the minor cycle where the mth occurrence of page O_{ijk} and it can be computed as follows.

$$M^m_{O_{ijk}} = j + \frac{LCM}{R_i} \times (m-1)\ .$$

For instance, for the example shown in Figure 3, page O_{231} occurs two times in a major cycle. First, it occurs in the $M^1_{O_{231}}$th minor cycle. Then, it occurs in the $M^2_{O_{231}}$th minor cycle, where

$$\begin{aligned} M^1_{O_{231}} &= 3 + \tfrac{6}{2} \times (1-1) = 3, \text{ and} \\ M^2_{O_{111}} &= 3 + \tfrac{6}{2} \times (2-1) = 6. \end{aligned}$$

Let $SN^m_{O_{ijk}}$ represent the corresponding sequence number of the mth occurrence of page O_{ijk}.

$$SN^m_{O_{ijk}} = \textstyle\sum_{p=1}^{M^m_{O_{ijk}}-1} MS_p + \sum_{p=1}^{i-1} NSC_{p,q} + k,$$

where $q = ((M^m_{O_{ijk}} - 1) \bmod NC_p) + 1$.

For instance, for the example shown in Figure 3, the two corresponding sequence numbers of page O_{231} are:

$$\begin{aligned} SN^1_{O_{231}} &= \textstyle\sum_{p=1}^{3-1} MS_p + \sum_{p=1}^{2-1} NSC_{p,q} + k \\ &= (MS_1 + MS_2) + NSC_{1,1} + 1 \\ &= (5+4) + 1 + 1 \\ &= 11. \\ SN^2_{O_{231}} &= \textstyle\sum_{p=1}^{6-1} MS_p + \sum_{p=1}^{2-1} NSC_{p,q} + k \\ &= (MS_1 + MS_2 + MS_3 + MS_4 + MS_5) \\ &\quad + NSC_{1,2} + 1 \\ &= (5+4+4+4+3) + 1 + 1 \\ &= 22. \end{aligned}$$

Let $SP^n_{O_{ijk}}$ denote the nth distance between the same page O_{ijk} in a major cycle and it can be computed as follows. When $n \neq R_i$, $SP^n_{O_{ijk}} = SN^{(n+1)}_{O_{ijk}} - SN^n_{O_{ijk}}$. When $n = R_i$, $SP^{R_i}_{O_{ijk}} = TS - SN^{R_i}_{O_{ijk}} + SN^1_{O_{ijk}}$.

Finally, we have

$$\begin{aligned} & EDP_{O_{ijk}} \\ &= (1/TS) \times \textstyle\sum_{m=1}^{R_k}((SP^m_{O_{ijk}} - 0.5) + \\ &\quad (SP^m_{O_{ijk}} - 1 - 0.5) + ... + \\ &\quad (SP^m_{O_{ijk}} - (SP^m_{O_{ijk}} - 1) - 0.5)) \\ &= (1/TS) \times \textstyle\sum_{m=1}^{R_k}({SP^m_{O_{ijk}}}^2/2) \\ &= (1/TS) \times (\textstyle\sum_{m=1}^{R_k-1}((SN^{m+1}_{O_{ijk}} - SN^m_{O_{ijk}})^2/2) + \\ &\quad (TS - SN^{R_i}_{O_{ijk}} + SN^1_{O_{ijk}})^2/2). \end{aligned}$$

Take page O_{231} (page 6) as an example, we have

$$\begin{aligned} & EDP_{O_{231}} \\ &= (1/23) \times ((\textstyle\sum_{m=1}^{2-1}(SN^{m+1}_{O_{231}} - SN^m_{O_{231}})^2/2) + \\ &\quad (23 - SN^2_{O_{231}} + SN^1_{O_{231}})^2/2) \\ &= (1/23) \times ((SN^2_{O_{231}} - SN^1_{O_{231}})^2/2 + \\ &\quad (23 - SN^2_{O_{231}} + SN^1_{O_{231}})^2/2) \\ &= (1/23) \times ((22-11)^2/2 + (23 - 22 + 11)^2/2) \\ &= (1/23) \times (10^2/2 + 12^2/2) \\ &= 1/23) \times (100/2 + 144/2) \\ &= 5.304. \end{aligned}$$

4.3. Simulation Results

In this simulation, we let $\theta = 0.8$, $\gamma = 0.9$. We consider 12 test samples which include the combinations of $S = 3$ and 4 and $\Delta = 4$ and 5, respectively, for a fixed D that is a random value between 10000 and 11000 to assure that $K_i \geq NC_i$. For each test sample, we compute the average result for 1000 values of D. The parameters and their default settings are shown in Table 3.

The total number of slots in a major cycle equals summing the total number of slots in each minor cycle. In the Acharya *et al.*'s algorithm, the size of each minor cycle equals MS $(= \sum_{i=1}^{S} NS_i = \sum_{i=1}^{S} \lceil \frac{K_i \times R_i}{L} \rceil)$. However, in the adaptive algorithm, the size of each minor cycle either equals MS or is less than MS. Therefore, obviously, the total number of slots in the adaptive approach is less than that in Acharya *et al.*'s algorithm.

Table 3: The parameters and their default settings

Parameter	*Default value*
S	3..4
D	10000..11000
Δ	4..5
θ	0.8
γ	0.9

Table 4: $\Delta = 4$, $R_i = (S - i) \times 4 + 1$, $1 \leq i \leq S$

S	*Adaptive*	BD	TWS *of* BD	$Max.\ TWS$
3	40144	40203	59	115
4	53874	54991	1117	2016

When $\Delta = 4$ and 5, the detailed simulation results about the total number of slots in one broadcast cycle in the adaptive approach and Acharya *et al.*'s algorithm for 1000 executions are shown in Tables 4 and 5, respectively, where *Adaptive* denotes the adaptive approach, BD denotes the Acharya *et al.*'s broadcast disk approach, TWS *of* BD denotes the total number of wasted slots in Acharya *et al.*'s broadcast disk approach, and $Max.\ TWS$ denotes the maximum number of wasted slots in Acharya *et al.*'s broadcast disk approach. From the results, we show that our adaptive approach always generates a smaller number of slots than Acharya *et al.*'s algorithm. As Δ is increased, the total number of slots is increased in both the adaptive approach and Acharya *et al.*'s algorithm. As S is increased, the total number of slots and the percentage of the total number of wasted slots are also increased in both the adaptive approach and Acharya *et al.*'s algorithm. When $S = 4$ and $\Delta = 4$, up to 2016 slots are empty among 54991 slots in Acharya *et al.*'s algorithm. (Note that as shown in Table 2, when $\Delta = 4$, it has the maximum value of LCM among the cases of $\Delta = 4$ and 5, since each R_i can not be divided by each other. But considering the results shown in Tables 4 and 5, a test sample with the maximum LCM ($\Delta = 4$) does not generate the maximum number of slots in one broadcast cycle; the maximum number of slots in one broadcast cycle occurs in the case of $\Delta = 5$.)

A comparison of the mean access time (in terms

Table 5: $\Delta = 5$, $R_i = (S - i) \times 5 + 1$, $1 \leq i \leq S$

S	*Adaptive*	*BD*	*TWS of BD*	*Max. TWS*
3	47555	47644	88	179
4	64716	65705	988	1750

Table 6: A comparison of the mean access time: Adaptive vs. BD

	Adaptive		*BD*	
S	$\Delta = 4$	$\Delta = 5$	$\Delta = 4$	$\Delta = 5$
3	2536.971	2888.545	2540.726	2893.961
4	2010.891	2311.529	2052.620	2346.846

of the time units) in the Acharya *et al.*'s algorithm and the adaptive approach for 1000 executions is shown in Table 6.

From this result, we show that the mean access time in our adaptive approach is always smaller than or equal to that in Acharya *et al.*'s algorithm. As S is increased, the access time is decreased in both the adaptive approach and Acharya *et al.*'s algorithm. As Δ is increased, the access time is increased in both the adaptive approach and Acharya *et al.*'s algorithm. When $S = 4$ and $\Delta = 4$, our adaptive approach can save 42 time units as compared with Acharya *et al.*'s algorithm. When $S = 4$ and $\Delta = 5$, our adaptive approach can save 35 time unit as compared with Acharya *et al.*'s algorithm.

5. Conclusion

Acharya *et al.* have proposed the use of a periodic dissemination architecture in the context of mobile systems, called Broadcast Disks. This strategy can construct a memory hierarchy in which the highest level contains a few items and broadcasts them with high frequency while subsequent levels contain more and more items and broadcast them with less and less frequency. In this way, one can establish a trade-off between access time for high-priority data and that of the low-priority items. However, based on Acharya *et al.*'s approach, some broadcast slots may be unused, which results in the waste of bandwidth and the increase of access time. In this paper, we have presented an adaptive approach to prevent the occurrence of the empty slots. Although there is a constraint, $K_i \geq NC_i$, the proposed approach guarantees no empty slots, which can save bandwidth. From our performance analysis and simulation, we have shown that our adaptive approach generates a smaller number of slots in one broadcast cycle and shorter mean access time than Acharya *et al.*'s algorithm. In an environment where different clients may listen to different number of broadcast channels, the schedules on different broadcast channels should be coordinated so as to minimize the access time for most clients [24]. Therefore, how to design efficient broadcast programs for the case of broadcasting over multiple channels is one of the possible future research directions.

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. on Management of Data*, pp. 199–210, 1995.

[2] S. Acharya, M. Franklin, and S. Zdonik, "Dissemination-Based Data Delivery Using Broadcast Disks," *Personal Comm.*, Vol. 2, No. 6, pp. 50–60, Dec. 1995.

[3] S. Acharya, M. Franklin, and S. Zdonik, "Prefetching from a Broadcast Disk," *Proc. of the 12th IEEE Int. Conf. on Data Eng.*, pp. 276–285, 1996.

[4] 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.

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

[6] T. F. Bowen, G. Gopal, G. Herman, T. Hickey, K. C. Lee, W. H. Mansfield, J. Raitz and A. Weinrib, "The Datacycle Architecture," *CACM*, Vol. 35, No. 12, pp. 71–81, Dec. 1992.

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

[8] 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 Eng.,* Vol. 18, No. 5, pp. 641–666, Sept. 2002.

[9] Y. I. Chang and W. H. Hsieh, “An Efficient Scheduling Method for Query-Set-based Broadcasting in Mobile Environments,” *Proc. of IEEE the 2nd Int. Workshop on Mobile Distributed Computing,* pp. 478–483, 2004.

[10] Y. C. Chehadeh, A. R. Hurson and M. Kavehrad, “Object Organization on a Single Broadcast Channel in the Mobile Computing Enviorment,” *Multimedia Tools and Applications*, Vol. 9, No. 1, pp. 69–94, July 1999.

[11] C. C. Chen, “Compression-based Broadcast Data for Reducing Access Time in Wireless Environment,” *Proc. of 1999 National Computer Symp.*, Vol. 3, pp. 539–546, 1999.

[12] M. S. Chen, K. L. Wu and P. S. Yu, “Optimizing Index Allocation for Sequential Data Broadcasting in Wireless Mobile Computing,” *IEEE Trans. on Knowledge and Data Eng.*, Vol. 15, No. 1, pp. 161–173, Jan./Feb. 2003.

[13] Y. D. Chung and M. H. Kim, “QEM: A Scheduling Method for Wireless Broadcast Data,” *Proc. of the 6th Int. Conf. on Database Systems for Advanced Applications*, pp. 135–142, 1999.

[14] Y. D. Chung, S. H. Bang and M. H. Kim, “An Efficient Broadcast Data Clustering Method for Multipoint Queries in Wireless Information Systems,” *The Journal of Systems and Software*, Vol. 64, No. 3, pp. 173–181, Dec. 2002.

[15] J. L. Huang, M. S. Chen and W. C. Peng, “Broadcasting Denpendent Data for Ordered Queries Without Replication in a Multi-Channel Mobile Environment,” *Proc. of the 19th Int. Conf. on Data Eng.*, pp. 692–693, 2003.

[16] J. L. Huang and M. S. Chen, “Dependent Data Broadcasting for Unordered Queries in a Multiple Channel Mobile Evironment,” *IEEE Trans. on Knowledge and Data Eng.*, Vol. 16, No. 9, pp. 1143–1156, Sept. 2004.

[17] T. Imielinski and S. Viswanathan, and B.R. Badrinath, “Data on Air: Organization and Access,” *IEEE Trans. on Knowledge and Data Eng.*, Vol. 9, No. 3, pp. 353–371, May/June, 1997.

[18] S. Jung, B. Lee and S. Pramanik, “A Tree-Structured Index Allocation Method with Replication over Multiple Broadcast Channels in Wireless Environments,” *IEEE Trans. on Knowledge and Data Eng.*, Vol. 17, No. 3, pp. 311–325, March 2005.

[19] J. Juran, A. R. Hurson, N. Yijaykrishnan and S. Kim, “Data Organization and Retrieval on Parallel Air Channels: Performance and Energy Issues,” *Wireless Networks*, Vol. 10, No. 2, pp. 183–195, March 2004.

[20] G. Lee and S. C. Lo, “Broadcast Data Allocation for Efficient Access of Multiple Data Items in Mobile Environments,” *Mobile Networks and Applications*, Vol. 8, No. 4, pp. 365–375, Aug. 2003.

[21] S. C. Lo and A. L. P. Chen, “An Adaptive Access Method for Broadcast Data Under an Error-Prone Mobile Environment,” *IEEE Trans. on Knowledge and Data Eng.*, Vol. 12, No. 4, pp. 609–620, July/Aug. 2000.

[22] 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.

[23] K. L. Tan, J. X. Yu, and P. K. Enk, “Supporting Range Queries in a Wireless Environment with Nonuniform Broadcast,” *Data and Knowledge Eng.*, Vol. 29, No. 2, pp. 201–221, Feb. 1999.

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

[25] X. Yang and A. Bouguettaya, “Adaptive Data Access in Broadcast-Based Wireless Enivornments,” *IEEE Trans. on Knowledge and Data Eng.*, Vol. 17, No. 3, pp. 326–338, March 2005.