

A Hybrid Approach to Query Sets Broadcasting Scheduling for Multiple Channels in Mobile Information Systems*

YE-IN CHANG AND SHIH-YING CHIU

Department of Computer Science and Engineering

National Sun Yat-Sen University

Kaohsiung, 804 Taiwan

E-mail: changyi@cse.nsysu.edu.tw

In this paper, we investigate the broadcasting scheduling over multiple channels for the case where a mobile client might need more than one page (i.e., a query set of data pages). Ke has proposed the *SNV* (Set-based version-Non-overlap Version) strategy for query set broadcast scheduling in multiple channels. In the *SNV* strategy, the data pages of the same query set are put together as much as possible, and the strategy tries to avoid scheduling two or more pages of one query set at the same time slot of different channels. However, there are two disadvantages with the *SNV* strategy: (1) a data page with high access frequency may be scheduled at a time slot near the end of the broadcast cycle, which will result in long access time for the whole query set; (2) it may increase the number of slots in a certain chain, which will result in a waste of bandwidth of the other channels. Therefore, in this paper, we propose an efficient broadcast scheduling strategy, a hybrid approach, to improve these two disadvantages. Basically, our hybrid approach combines the advantages of page-based and set-based strategies. Through our performance analysis and simulation, we show that our hybrid approach requires less total expected delay access time and creates a smaller number of slots and a smaller number of empty slots in one broadcast cycle compared to the *SNV* strategy.

Keywords: access time, bandwidth, broadcast schedule, mobile information systems, multiple channels

1. INTRODUCTION

The emergence of powerful portable computers, along with advances in wireless communication technologies, has made mobile computing a reality [8, 17]. In the evolving field of mobile computing, there is a growing concern to provide mobile users with timely access to large amounts of information [20, 34]. Examples of such services include weather, highway conditions, traffic directions, news and stock quotes. Due to the intrinsic constraints of mobility, such as bandwidth restrictions, power consumption and connection reliability, designing an efficient, scalable and cost effective mobile information access system is a challenging task.

Although a wireless network with mobile clients is essentially a distributed system, there are some characteristic features that make the system unique and a fertile area of research [8]. These features include *asymmetry in communications*, *frequent discon-*

Received August 30, 2001; accepted April 15, 2002.

Communicated by Jang-Ping Sheu, Makoto Takizawa and Myongsoon Park.

* 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.

nections, power limitations and screen size. Each one of these features has an impact on how data can be effectively managed in a system with MCs (Mobile Clients) [8]. Among them, asymmetry in the communications means that the bandwidth in the downstream direction (servers-to-clients) is much greater than that in the upstream direction. The communication asymmetry, along with the restriction in power that mobile units have, make the model of broadcasting data to client, an attractive proposition.

Basically, in a wireless environment, servers can deliver data to clients in two modes [28]: (1) Broadcasting Mode: Data is broadcast periodically on a downlink channel. (2) On-Demand Mode: Clients submit their requests on an uplink channel, and servers respond by sending data to clients on a downlink channel.

Although the on-demand mode lets mobile clients play a more active role in getting the data they need, it is not suitable for asymmetric communication environments on account of upstream its narrower communication capabilities. This restriction means that only some clients can be served in the on-demand mode. On the other hand, with the broadcasting mode, the number of mobile clients which can simultaneously listen to the downlink channel can be almost infinitely scaled. In a wireless computing environment, using the broadcasting mode not only can save bandwidth of the uplink channel, but also allows scaling to any number of mobile clients who listen to the published report.

Under the broadcasting mode, the server must construct a broadcast "program" to meet the needs of the client population [1]. Information broadcast by the server is organized into units called *pages*. Time on the broadcast channel is divided into slots of the same size, which is equal to the time needed to broadcast a page. The amount of time a client has to wait for an information item that it needs is called the *access time*. The access time depends on the broadcast schedule. A good broadcast program can minimize the mean access time.

Many strategies have been proposed for efficient broadcast delivery [5, 6, 11, 13, 15, 16, 23, 25-27]. Basically, these strategies can be classified into two types: *static* and *dynamic*. In static scheduling strategies, the number of objects and the set of objects in the publication group are predetermined and fixed. In contrast, in dynamic scheduling strategies, the number of objects and the set of objects in the publication group can be varied at runtime [28]. Also, some approaches reduce the access time by caching [2, 3, 7, 18, 33], and nonuniform broadcasting [3, 18, 30, 31], and some reduce the *tuning time* by indexing [14, 19-21, 29, 33, 35], hashing [19] or using signature [24] techniques, where the tuning time is the amount of time spent by a client listening to the channel. Strategies presented in [9, 10] consider real-time, fault-tolerant, secure broadcast organization techniques. Strategies presented in [30, 31] generate broadcast programs that facilitate range queries for multiple-disk broadcast programs.

However, most of the previous approaches assume that each mobile client needs only one data page. In many situations, a mobile client might need data from more than one page. For example, users often need prices of one or more stocks or traffic information about several roads. Under these circumstances, minimizing the access time for mobile clients is a non-trivial task. The scheduling strategy should take the affinity between data pages into consideration [22]. The way a mobile client accesses a broadcast stream is illustrated in Fig. 1, where the server broadcasts a set of data objects $\{d_1, d_2, d_3, d_4, d_5\}$ in one *bcast*, and a client issues a query retrieving d_1 and d_4 [12]. (In this work, we assume there is no precedence relation in accessing data objects.) In Fig. 1, the client has to wait for the next *bcast* to access d_1 because the client's query is issued. However, if the broadcast schedule is formed as $\langle d_2, d_3, d_1, d_4, d_5 \rangle$, then the client can

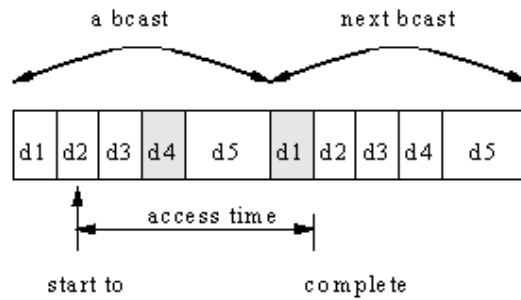


Fig. 1. A query contains a set of data objects.

retrieve $d1$ and $d4$ in current *bcast*. This means that an efficient schedule provides access time reduction to mobile clients.

Moreover, all of the previous works discussed how to broadcast data over single channels. In [22], Ke studied the issue of scheduling broadcast data on multiple wireless channels as an MC may access more than one data page. That author also proposed page-based scheduling strategies and set-based scheduling strategies. In paged-based scheduling strategies, that is, the *PHV* strategy and the *PVV* strategy, a data page with the maximal access frequency is broadcast first. Therefore, clients can retrieve the desired data pages as fast as possible [22]. However, both the *PHV* strategy and the *PVV* strategy ignore the relationship between data pages. Therefore, a conflict between pages P_i and P_j can occur, where pages P_i and P_j are in the same query set and are broadcast at the same time slot of different channels. Since we cannot retrieve pages P_i and P_j in the same broadcast cycle, this will result in an increase of the total access time. In the set-based strategies, that is, the *SHV* strategy and the *SVV* strategy, the pages of the same query set are put together as much as possible. Moreover, data pages of the same query set are assigned to different time slots to avoid conflicts. However, neither the *SHV* strategy nor the *SVV* strategy can guarantee that there will be no conflict between data pages. In order to reduce the probability of conflicts between data pages, Ke proposed another strategy, the *SNV* (Set-based strategy-Non-overlap Version) strategy [22]. The basic idea of the *SNV* strategy is to assign all query sets to the available channels in a round-robin pattern and try to avoid scheduling two or more pages of one query set at the same time slot of different channels. However, a data page with high access frequency may be scheduled at a time slot near the end of the broadcast cycle. That will result in an increase of the total access time. Moreover, when all the *predictive* slots in the broadcast scheduling matrix cause a conflict for a certain page, that page will be assigned to the extended slot of the last channel, where the value of the *predictive* slots is equal to $\lceil \frac{\text{the number of data pages}}{\text{the number of channels}} \rceil$. This will result in wasted bandwidth.

Therefore, we propose an efficient broadcast scheduling strategy, a hybrid approach designed to improve these two above disadvantages. Basically, our hybrid approach combines the advantages of the page-based and set-based strategies. The basic idea is to simultaneously consider the request ratios of query sets and the access frequencies of pages. The query set with the maximal request ratio is assigned to the broadcast matrix first, while for pages in a query set, the page with the maximal access frequency is assigned to the broadcast matrix first. Query sets are assigned to broadcast channels in top-down order, while pages in a query set are assigned to the broadcast channels in right-left-order.

The rest of this paper is organized as follows. In section 2, we give a brief survey of Ke's *SNV* strategy and show two examples to illustrate the disadvantages of Ke's strategy. In section 3, we present our hybrid approach. In section 4, we study the performance of our hybrid approach and compare it with Ke's *SNV* strategy. Finally, section 5 gives a conclusion.

2. BACKGROUND

In this section, we first describe Ke's *SNV* strategy. Next, we give two examples to show the disadvantages of this strategy.

2.1 The *SNV* Strategy

The basic idea of the *SNV* strategy is to assign all data sets to the available channels in a round-robin manner. Moreover, the *SNV* strategy also tries to avoid scheduling two or more data pages of one data set at the same time slot of different channels. For the example shown in Fig. 2, Fig. 3 shows the resulting broadcast matrix (M) obtained using the *SNV* strategy, and Fig. 4 shows the expected access time of each set.

Query set	Data pages	Request ratio (%)
Q_1	P_1, P_2, P_3, P_4	35
Q_2	P_3, P_6	25
Q_3	P_4, P_7, P_8	20
Q_4	P_2, P_9, P_{11}, P_{14}	8
Q_5	$P_6, P_{10}, P_{12}, P_{14}$	7
Q_6	$P_5, P_{11}, P_{13}, P_{15}$	5

Fig. 2. Example 1 (adapted from [22]).

$$m = \begin{bmatrix} P_1 & P_2 & P_3 & P_4 & P_9 \\ P_6 & P_{10} & P_{11} & P_{14} & P_{12} \\ P_7 & P_8 & & P_5 & P_{13} & P_{15} \end{bmatrix}$$

Fig. 3. The output matrix of example 1 based on the *SNV* strategy.

Query set	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6
Access time	4	3	4	5	5	6

Fig. 4. The expected access time of each query set for Example 1 based on the *SNV* strategy.

Now, we will discuss how the data pages of query set Q_4 are scheduled. (Note that before pages in query set Q_4 can be scheduled, pages $P_1, P_2, P_3, P_4, P_6, P_7$ and P_8 have to be scheduled.) First, page P_2 is already scheduled since it is also included in query set Q_1 . Next, page P_9 is assigned to slot 5 of channel 1 due to $4 \pmod 3$ (the query set number) mod 3 (the total number of channels) = 1. In this example, the number of available

channels is 3, and the total number of pages is 15, so the *predictive* number of slots in the channel is $\lceil 15/3 \rceil = 5$. Therefore, the other data pages of query set Q_4 should be broadcast on the next channel, i.e., channel 2. Because page P_2 belongs to query set Q_4 and has been assigned to slot 2 of channel 1, slot 2 of channel 1 will cause page P_2 to conflict with other data pages in query set Q_4 . Hence, pages P_{11} and P_{14} are assigned to slot 3 and slot 4 of channel 2, respectively, instead of to slot 2 of channel 2. At this point, pages in query set Q_4 have all been scheduled. Next, page P_{12} in query set Q_5 is assigned to slot 5 of channel 2. Finally, pages P_5, P_{13} and P_{15} are assigned to slots 4, 5 and 6 of channel 3, respectively. (Note that slot 3 of channel 3 can not be assigned to any page in query set Q_6 since page P_{11} , which is assigned to slot 3 of channel 2, belongs to query set Q_6 .) The expected access time is calculated by multiplying the probability of access for each set (R_i) by the expected delay for that set (AT_i) and summing the results. Therefore, the expected the access time of this example to be as follows:

$$\sum_{i=1}^{NQ} R_i \times AT_i = \sum_{i=1}^6 R_i \times AT_i = (4 \times 35 + 3 \times 25 + 4 \times 20 + 5 \times 8 + 5 \times 7 + 6 \times 5) / 100 = 4.$$

Data pages	P_3	P_4	P_2	P_1	P_6	P_7	P_8	P_{14}	P_{11}	P_9	P_{10}	P_{12}	P_5	P_{13}	P_{15}
Access frequency	60	55	43	35	32	20	20	15	13	8	7	7	5	5	5

Fig. 5. The data pages of example 1 after sorting on their frequencies.

2.2 Two Bad Examples of the SNV Strategy

Given six query sets and their request ratios as shown in Fig. 2, Fig. 3 shows the result based on the *SNV* strategy. Note that, based on the access frequency of each page as shown in Fig. 5, page P_3 has the highest access frequency; however, it is assigned to slot 3 of channel 1 instead of to slot 1 of channel 1. Therefore, when query set Q_2 is retrieved, it takes 3 time units to read the data. If we assign page P_3 to slot 1 of channel 1 and assign page P_6 to slot 2 of channel 2, then it takes only 2 time units to retrieve query set Q_2 . Note that when page P_3 is assigned to slot 1 of channel 1, the access time of query set Q_1 is not affected, but that of query set Q_2 is decreased. In the *SNV* strategy, the pages in a query set are assigned to the broadcast matrix in random order, so a data page with high access frequency may be scheduled at the time slot near the end of the broadcast cycle. This will result in an increase of the total access time.

Query set	Data pages	Request ratio (%)
Q_1	P_1, P_2, P_3, P_4, P_5	35
Q_2	$P_1, P_2, P_6, P_{10}, P_{12}$	25
Q_3	P_1, P_3, P_4, P_8, P_9	20
Q_4	P_2, P_7, P_{11}	8
Q_5	P_1, P_8, P_{13}, P_{15}	7
Q_6	P_1, P_8, P_{14}	5

Fig. 6. Example 2.

Let us examine another input example, shown in Fig. 6. In this example, the number of available channels is 3, and the total number of pages is 15, so the *predictive*

1. D : the number of pages;
2. NQ : the number of query sets;
3. Q_i : the i th query set of data pages, $1 \leq i \leq NQ$;
4. R_i : the request ratio of query set Q_i , $1 \leq i \leq NQ$;
5. P_i : the i th page, $1 \leq i \leq D$;
6. AF_i : the access frequency of page P_i , $1 \leq i \leq D$;
7. F_i : the flag to indicate whether a page P_i has been assigned in the broadcast schedule, $1 \leq i \leq D$; if $f_i = 0$, then this indicates that page P_i has not been assigned in the schedule; otherwise, page P_i has been assigned in the schedule;
8. X : a set containing all the pages;
9. TQ_i : the temporal query list of query set Q_i , $1 \leq i \leq NQ$;
10. AC : the number of available channels;
11. SC : the number of available slots per channel on average; its initial value is computed as $SC = \left\lceil \frac{D}{AC} \right\rceil$;
12. $M[i, j]$: a broadcast matrix, where the value of $M[i, j]$ is the data page broadcast at the j th time slot of the i th channel, $1 \leq i \leq AC$, $1 \leq j \leq SC$;
13. NS_i : the next available slot of channel i , $1 \leq i \leq AC$;
14. AS_i : a list of available slots before NS_i , $1 \leq i \leq AC$;
15. C : the number of channels which have been checked for a page;
16. ROW : the ROW 'th row of the matrix $M[i, j]$;
17. COL : the COL 'th column of the matrix $M[i, j]$;
18. $TASE$: an element in list AS_i .

3.3 Basic Ideas

The proposed *Hybrid* strategy combines the advantages of the page-based and the set-based strategies and also improves the two disadvantages of the *SNV* strategy. In the page-based strategy, the page with the highest access frequency is assigned to the broadcast matrix first. In the set-based strategy, the query set with the highest request ratio is assigned to the broadcast matrix first. The basic idea of the hybrid approach is to consider the request ratios of query sets and the access frequencies of pages at the same time. The query set with the highest request ratio is assigned to the broadcast matrix first. While for pages in a query set, the page with the highest access frequency is assigned to the broadcast matrix first. Query sets are assigned to the broadcast channels from top to down, while pages in a query set are assigned to the broadcast channels from left to right.

$$m = \begin{bmatrix} P_3 & P_4 & P_2 & P_1 & P_{14} & \\ P_{11} & P_6 & P_{10} & P_9 & & \\ P_7 & P_5 & P_8 & P_{12} & P_{13} & P_{15} \end{bmatrix}$$

Fig. 8. The output matrix of Example 1 based on the hybrid approach.

For the same input example shown in Fig. 2, Fig. 8 shows the result based on the hybrid approach. Given $NQ = 6$, $D = 15$, and $AC = 3$, we have $SC = \lceil D/AC \rceil = 5$. In our

Query set	Data pages
TQ_1	$P_3, P_4, P_2, P_1,$
TQ_2	P_3^*, P_6
TQ_3	$P_4^*, P_7, P_8,$
TQ_4	$P_2^*, P_{14}, P_{11}, P_9$
TQ_5	$P_6^*, P_{14}^*, P_{10}, P_{12}$
TQ_6	$P_{11}^*, P_5, P_{13}, P_{15}$

P_i^* : the repeating occurrence of page P_i .

Fig. 9. TQ_i in Example 1 ($1 \leq i \leq NQ$).

approach, first, we calculate the access frequency of each of these 15 pages. For this example, the result is shown in Fig. 5. Next, we sort the pages (P_i) in each query set (Q_i) on their access frequencies (AF_i) in decreasing order, and assign the result to a temporary query list (TQ_i) as shown in Fig. 9. Then, we assign each query list TQ_i to the broadcast channels in round-robin fashion. That is, query lists $TQ_1, TQ_2, TQ_3, TQ_4, TQ_5$ and TQ_6 are assigned to channels 1, 2, 3, 1, 2 and 3, respectively. Here, we need two extra variables: NS_i and AS_i , $1 \leq i \leq AC$. NS_i denotes the next available slot of channel i and AS_i denotes a list of available slots before NS_i .

For pages in query list TQ_1 , page P_3 has the highest access frequency, so it is assigned to the broadcast matrix first. Consequently, pages P_3, P_4, P_2 , and P_1 are assigned to slots 1 to 4 of channel 1, respectively. The next available slot in channel 1 is slot 5, i.e., $NS_1 = 5$. At this point, all the pages in query list TQ_1 have been assigned to slots. Next, pages P_3 and P_6 in query list TQ_2 are considered. Basically, each data page is broadcast once within each broadcast cycle, so we will not process page P_3 in query list TQ_2 again. We assign page P_6 in query list TQ_2 to slot 2 of channel 2, instead of to slot 1 of channel 2, to avoid conflicting with page P_3 , and we have $NS_2 = 3$. Moreover, we record slot 1 in list AS_2 (i.e., the list of the available slots before NS_2). Here, we have to know how to check whether a slot is a conflicting slot for a page P_k . If the slot in any available channel contains a page which is in the same query set as page P_k , then that slot is a conflicting slot for page P_k . For the previous example, after query list TQ_1 is assigned to the broadcast matrix, slot 1 is a conflicting slot for page P_6 in query list TQ_2 since slot 1 of channel 1 contains page P_3 , which is also in the same query list TQ_2 . For pages in query list TQ_3 , page P_4 is ignored for the same reason as page P_3 , and pages P_7 and P_8 are assigned to slot 1 and slot 3 of channel 3, respectively. Therefore, we have $NS_3 = 4$ and list $AS_3 = \langle 2 \rangle$.

So far, we have considered all three channels. Therefore, when we handle the case of query list TQ_4 , we re-consider channel 1 again. (Note that page P_2 is ignored for the same reason as page P_3 .) Since $NS_1 = 5$, page P_{14} in query list TQ_4 is assigned to slot 5 of channel 1 and NS_1 is updated to 6. At this point, pages P_{11} and P_9 in query list TQ_4 have not been assigned to the broadcast matrix, and slot 6 ($= NS_1$) is out of the range of the slots of channel 1 ($NS_1 (= 6) > SC (= 5)$). Therefore, we go down to the next channel, i.e., channel 2. At this time, since list AS_2 is not empty, we will try the slot(s) in list AS_2 first, instead of trying slot NS_2 . Therefore, although $NS_2 = 3$, page P_{11} is assigned to slot 1, instead of slot 3, of channel 2. This is because slot 1 recorded in list AS_2 will not cause page P_{11} to conflict with another page. Note that to reduce the total access time for pages in a query set, we should apply the following policies from the high priority to

the low priority to decide on a *good* slot for a page: (1) if $|AS_{ROW}| \neq 0$, we try to find a non-conflicting slot in AS_{ROW} ; (2) if there is no non-conflicting slot in AS_{ROW} , we try to find a non-conflicting slot from slot NS_{ROW} ; (3) if we can not find a non-conflicting slot based on the above two policies, we go down to the next channel. As for page P_9 in query list TQ_4 , it is assigned to slot 4, instead of to slot 3, of channel 2, to avoid conflicting with page P_2 (in the same query set Q_4). Finally, we update NS_2 as $NS_2 = 5$.

When query list TQ_5 is considered, pages P_6 and P_4 are ignored for the same reason as page P_3 . Therefore, only pages P_{10} and P_{12} are considered. At this point, slots of channel 2 are considered due to 5 (i.e., the identifier of the query list TQ_5) mod 3 ($= AC$) = 2. Page P_{10} in query list TQ_5 is assigned to slot 3 ($\in AS_2$) of channel 2 for the same reason of page P_{11} in query list TQ_4 , and we update $AS_2 = AS_2 - \langle 3 \rangle$. For page P_{12} , slot 5 ($= NS_2$) in channel 2 will cause page P_{15} to conflict with page P_{14} , so we record slot 5 in list AS_2 and let $NS_2 = 6$. However, slot 6 ($= NS_2$) is out of the range of slots of channel 2. Therefore, we go down to channel 3. Although we want to assign a page to a slot near the beginning of the broadcast cycle, slot 2 in the set AS_3 will cause page P_{12} to conflict with page P_6 (in the same query list TQ_5). Therefore, page P_{12} is assigned to slot 4 ($= NS_3$), instead of to slot 2, of channel 3, and we update NS_3 as $NS_3 = 5$.

For pages in query list TQ_6 , page P_{11} is ignored for the same reason as page P_3 . Therefore, only pages P_5 , P_{13} and P_{15} are considered. Moreover, since $6 \text{ mod } 3 = 0$, channel 3 is considered. First, at this point, page P_5 is assigned to slot 2 ($\in AS_3$) of channel 3 for the same reason as page P_{11} in query list TQ_4 . Next, page P_{13} is assigned to slot 5 ($= NS_3$) of channel 3, and we update $NS_3 = 6$. After that, we have to consider page P_{15} . However, since list AS_3 is empty and slot 6 ($= NS_3$) is out of the range of the slots of channel 3, we go to the next channel, i.e., channel 1. For the same reason ($AS_1 = \langle \rangle$ and $NS_1 = 6$), we go to channel 2. In channel 2, slot 5 ($\in AS_2$) will cause page P_{15} to conflict with page P_{13} and $NS_2 (= 6) > SC (= 5)$, so we can not assign page P_{15} to channel 2. We then go down to channel 3 again (which is detected by recording the number of channels that have been gone through). Since there is no non-conflicting slot in any of the three channels for page P_{15} , we can only assign page P_{15} to the extended slot of channel 3, i.e., slot 6 of channel 3. Finally, since one extended slot has been used, we extend the size of the broadcast matrix to 3×6 . Fig. 8 shows the result based on the hybrid approach. Fig. 10 and Fig. 11 show the expected access time of each query set using the hybrid approach and the SNV strategy, respectively. The total expected access time, which is denoted as $TEAT$, is calculated by multiplying the request ratio of each data set (R_i) by the access time for that data set (AT_i) and summing the results. That is,

$$TEAT = \sum_{i=1}^{NQ} R_i \times AT_i.$$

Query set	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6
Access time	4	2	3	5	5	6

Fig. 10. The expected access time of Example 1 based on the hybrid approach.

Query set	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6
Access time	4	3	4	5	5	6

Fig. 11. The expected access time of Example 1 based on the SNV strategy.

The total expected access time based on the hybrid approach is as follows:

$$\begin{aligned} TEAT &= \sum_{i=1}^6 R_i \times AT_i \\ &= (4 \times 35 + 2 \times 25 + 3 \times 20 + 5 \times 8 + 5 \times 7 + 6 \times 5) / 100 = 3.55. \end{aligned}$$

The total expected access time based on the SNV strategy is as follows:

$$\begin{aligned} TEAT &= \sum_{i=1}^6 R_i \times AT_i \\ &= (4 \times 35 + 3 \times 25 + 4 \times 20 + 5 \times 8 + 5 \times 7 + 6 \times 5) / 100 = 4. \end{aligned}$$

Consequently, the total expected access time of the hybrid approach is less than that of the SNV strategy.

3.4 The Hybrid Algorithm

Given AC channels and NQ query sets Q_1, Q_2, \dots, Q_{NQ} with request ratios R_1, R_2, \dots, R_{NQ} , respectively, we have to assign all pages to the broadcast matrix, which is an $AC \times SC$ matrix. The complete algorithm as shown in Fig. 12 contains the following steps:

1. Call procedure *HybridInitialization* (as shown in Fig. 13) to initialize the values of variables. In the *HybridInitialization* procedure, first, we calculate the number of available slots per channel on average, which is denoted as SC . Second, we order the query sets (Q_i) from hottest to coldest based on their request ratios. Third, we compute the access frequency (AF_i) for each page (P_i) and set F_i to 0, which indicates that no pages have been assigned to the broadcast matrix. Fourth, order the pages in each query set on their access frequency (AF_i) in a decreasing order and assign the result to a temporary query list, TQ_i . Finally, we initialize the values of some variables: AS_i, NS_i, ROW and $COL, 1 \leq i \leq AC$.
2. For each query list TQ_r , the pages ($\in TQ_r$) are assigned to the broadcast matrix in the following manner:


```

for r := 1 to NQ do
begin
  while ( $|TQ_r| \neq 0$ ) do
  begin
    select the first page  $P_k$  from  $TQ_r$ ;
     $TQ_r := TQ_r - \langle P_k \rangle$ ;
    if the page  $P_k$  has not been assigned to the broadcast matrix then
      call procedure AssignPosition (as shown in Fig. 14)
      to assign it to the broadcast matrix;
  end;
end.

```

```

01 Input: The number of available channels, which is denoted as  $AC$ , and  $NQ$  data sets  $Q_1, Q_2, \dots, Q_{NQ}$  with request ratios  $R_1, R_2, \dots, R_{NQ}$ , respectively;
02 Output:  $M[i, j]$ , a broadcast matrix,  $1 \leq i \leq AC, 1 \leq j \leq SC$ ;
03 begin
04   Call HybridInitialization;
05   For  $r := 1$  to  $NQ$  do
06     begin
07       While ( $|TQ_r| \neq 0$ ) do
08         begin
09           Select the first page  $P_k$  from  $TQ_r$ ;
10            $C := 0$ ;
11            $TQ_r := TQ_r - \langle P_k \rangle$ ;
12           while ( $F_k = 0$ ) do
13             call AssignPosition( $P_k, Q_r, ROW$ );
14           end;
15           If  $((r + 1) \bmod AC) = 0$  then  $ROW := 3$ 
16           else  $ROW := ((r + 1) \bmod AC)$ ;
17         end;
18     end.

```

Fig. 12. The Hybrid algorithm.

```

01 Procedure HybridInitialization;
02 begin
03    $SC := \left\lceil \frac{D}{AC} \right\rceil$ ;
04   Sort  $Q_i$  based on their request ratios  $R_i$  in decreasing order and reassign their indexes;
05   for  $i := 1$  to  $D$  do
06     begin
07        $F_i := 0$ ;
08        $AF_i := 0$ ;
09       for  $j = 1$  to  $NQ$  do
10         if  $P_i \in Q_j$  then  $AF_i := AF_i + R_j$ ; (* compute the access frequency for  $P_i$  *)
11       end;
12     for  $i := 1$  to  $NQ$  do
13       Sort  $P_k$  in  $Q_i$  on  $AF_k$  in decreasing order and assign the result to  $TQ_i$ ;
14      $X := \bigcup_i Q_i = \{P_1, P_2, \dots, P_n\}$  be all broadcast data pages;
15     for  $i := 1$  to  $AC$  do
16       begin
17          $NS_i := 1$ ;
18          $AS_i := \langle \rangle$ ;
19       end;
20      $ROW := 1$ ;
21      $COL := 1$ ;
22   end;

```

Fig. 13. The *HybridInitialization* procedure.

In Fig. 12, line 4 indicates what we do in step 1. Lines 5 to 17 indicates what we do in step 2. In step 2, we assign all input query sets to the broadcast matrix in round-robin fashion. (Line 15 and line 16 are used for this purpose.) The query set with the highest request ratio is assigned to the broadcast matrix first. Since we sort query sets based on their request ratios in step 1, the first query set has the highest request ratio, while the last query set has the lowest request ratio. For the previous example, query set Q_1 is assigned to the broadcast matrix first, while query set Q_6 is assigned to the broadcast matrix last. For pages in a query set, the page with the highest access frequency is assigned to the broadcast matrix first. Since we sort pages in each set based on their access frequencies and assign the result to a query list in step 1, the first element in a query list has the highest access frequency, while the last element has the lowest access frequency. For pages in query list TQ_1 , the first element, page P_3 , is assigned to the broadcast matrix first, while the last element, page P_1 , is assigned to the broadcast matrix last. For each page in each query list, if it has not been assigned to the broadcast matrix, we call procedure *AssignPosition* as shown in Fig. 14 to assign it to the broadcast matrix.

The purpose of procedure *HybridInitialization* shown in Fig. 13 is to initialize the values of variables. First, in line 3, we compute the predictive number of slots per channel based on the average ($SC = \left\lceil \frac{D}{AC} \right\rceil$). (Note that SC can be increased later.) In the

previous example as shown in Fig. 2, we have $SC = \left\lceil \frac{15}{3} \right\rceil = 5$. Next, in line 4, we order the query sets from hottest to coldest based on their request ratios, and the result is shown in Fig. 2. Third, in lines 5 to 11, we have to initialize the variables for each page.

In line 7, F_i is set to be 0, which indicates that no pages have been assigned to the broadcast matrix. From lines 8 to 10, we compute the access frequency of each page. For a page P_k , its access frequency (AF_k) is equal to the sum of the request ratios (R_i) of the query sets in which page P_k occurs. For example, page P_3 occurs in query sets Q_1 and Q_2 , so we have $AF_3 = R_1 + R_2 = 35 + 25 = 60$. The result of computing the access frequency of each page is shown in Fig. 5. Fourth, in lines 12 and 13, we order the pages in each query set (Q_i) from hottest to coldest and assign the result to the corresponding temporary query list, TQ_i . For our previous example, the result of TQ_i is shown in Fig. 9. Finally, in lines 15 to 21, we have to initialize the variables for each channel, ROW and COL . In line 17, NS_i is set to be 1, which indicates that the next available slot of channel i is slot 1. In line 18, AS_i is set to be an empty list, which indicates that there is no conflicting slot before NS_i in channel i . In lines 20 and 21, ROW and COL are set to be 1, which indicates that the first page should be assigned to the first slot of the first channel.

The purpose of procedure *AssignPosition*(P_k, Q_r, ROW) is to find a non-conflicting slot in row ROW of the broadcast matrix for a page P_k in query set Q_r . In procedure *AssignPosition*, the value of NS_{ROW} represents the next available slot in channel ROW , and there may exist a non-conflicting slot for P_k in slots NS_{ROW} to SC . In addition to those slots after NS_{ROW} (including NS_{ROW} , there may exist a non-conflicting slot in AS_{ROW} for page P_k . That is, for each channel, the non-conflicting slot may be in AS_{ROW} or after slot NS_{ROW} . However, we check the slots in AS_{ROW} first if list AS_{ROW} is not empty. This is because we prefer to assign a page to a slot near the beginning of the broadcast cycle. If there is no non-conflicting slot in AS_{ROW} for page P_k , then we check the slots in NS_{ROW} . Moreover, if we still can not find a non-conflicting slot for page P_k in slots NS_{ROW} to SC , we go down to check the slots in the next channel. If there is no non-conflicting slot in

```

01 Procedure AssignPosition( $P_k$  : integer;  $Q_r$  : set of pages; var  $ROW$  : integer);
02 begin
03    $u := TryAvailSlot(ROW, Q_r)$ ; (* Step A *)
04   If  $u \neq -1$  then (* Step B *)
05     begin (* there exists a non-conflicting slot before  $NS_{ROW}$  *)
06        $M[ROW, u] := P_k$ ;
07        $F_k := 1$ ;
08        $AS_{ROW} := AS_{ROW} - \{u\}$ ;
09     end;
10   If  $F_k = 0$  then (* all slots are conflicting before slot  $NS_{ROW}$  *)
11     begin
12       If  $NS_{ROW} > SC$  then (* Step C *)
13         begin (* Case 1: all slots of channel  $ROW$  have been checked *)
14            $ROW := (ROW \bmod AC) + 1$ ; (* go down *)
15            $C := C + 1$ ;
16         end
17       else (* Case 2: not all slots of channel  $ROW$  have been checked *)
18         begin
19            $COL := NS_{ROW}$ ; (* Step D *)
20           while (CheckConflict( $COL, Q_r$ )) and ( $COL \leq SC$ ); (* Step E *)
21             begin (* the slot  $COL$  will induce a conflict *)
22                $AS_{ROW} := AS_{ROW} + \{COL\}$ ; (* record the available slot *)
23                $COL := COL + 1$ ; (* go right *)
24             end;
25           If  $COL > SC$  then (* Step F *)
26             begin (* out of the range of slots of the  $ROW$ 'th channel*)
27                $NS_{ROW} := SC + 1$ ;
28                $ROW := (ROW \bmod AC) + 1$ ; (* go down *)
29                $C := C + 1$ ;
30             end
31           else (* Step G *)
32             begin (* a conflict does not occur *)
33                $M[ROW, COL] := P_k$ ;
34                $F_k := 1$ ;
35                $NS_{ROW} := COL + 1$ ;
36             end;
37           end; (* end of else *)
38         end; (* end of If  $F_k = 0$  *)
39       If ( $F_k = 0$ ) and ( $C \geq AC$ ) then (* Step H *)
40         begin (* all the channels have been checked *)
41            $COL := NS_{AC}$ ;
42            $SC := SC + 1$ ;
43            $M[AC, COL] := P_k$ ;
44            $F_k := 1$ ;
45            $NS_{AC} := NS_{AC} + 1$ ;
46         end;
47     end;

```

Fig. 14. The *AssignPosition* procedure.

any available channel, we assign page P_k to the extended slot, slot $(SC + 1)$, of the last channel, and update $SC = SC + 1$ to extend the broadcast matrix.

In procedure *AssignPosition*, we call function *TryAvailSlot* and function *CheckConflict* as shown in Fig. 15 and Fig. 16, respectively. The purpose of function *TryAvailSlot* (ROW, Q_r) is to check whether there is a non-conflicting slot in AS_{ROW} for the incoming page in Q_r . If function *TryAvailSlot* returns -1, this indicates that there is no non-conflicting slot in AS_{ROW} ; otherwise, the value of function *TryAvailSlot* is the non-conflicting slot number to which page $P_k (\in Q_r)$ can be assigned. The purpose of function *CheckConflict* (COL, Q_r) is to check whether an input slot, slot COL , is a conflicting slot for a page $P_k (\in Q_r)$. If function *CheckConflict* returns *True*, the input slot, slot COL , is a conflicting slot for page P_k ; otherwise, the input slot is not a conflicting slot.

In procedure *AssignPosition* (P_k, Q_r, ROW), any input page P_k goes through step *A* to try to find a non-conflicting slot in list AS_{ROW} first. Next, an input page P_k goes through step *B* only if there is a non-conflicting slot in AS_{ROW} ; otherwise, it goes through step *C* or step *D*. An input page P_k goes through step *C* only if slot NS_{ROW} is out of the range of slots of channel ROW ; otherwise, it will go through step *D*. Then, an input page P_k goes through step *E* if the slot at which it is going to be assigned is a conflicting slot for page P_k . After processing step *E*, we find a non-conflicting slot. Furthermore, if the non-conflicting slot which we find in step *E* is out of the range of slots of channel ROW , we process step *F*; otherwise, we perform step *G*. Finally, if there is no non-conflicting slot for page P_k in all available channels, we perform step *H*.

```

01 Function TryAvailSlot (ROW: integer; Qr : set of pages): integer;
02 begin
03   p := first (ASROW); (* p is the pointer to the list *)
04   TryAvailSlot := -1;
05   while (p ≠ nil) and (TryAvailSlot = -1) do
06     begin
07       Let TASE be the pth element of ASROW;
08       If (CheckConflict (TASE, Qr) = False) then
09         begin
10           TryAvailSlot := TASE;
11           ASROW := ASROW - <TASE> ;
12         end;
13       p := next (ASROW) ;
14     end;
15 end;
```

Fig. 15. The *TryAvailSlot* function.

The purpose of function *CheckConflict* (COL, Q_r) is to check whether slot COL is a conflicting slot for a page $P_k (\in Q_r)$. If slot COL in any available channel contains a page which is in the same query set Q_r as page P_k , then slot COL is a conflicting slot for page $P_k (\in Q_r)$, and the resulting value of function *CheckConflict* is *True*. For the previous example, after assigning all the pages in query set Q_1 to the broadcast matrix, we want to check whether slot 1 is a conflicting slot for page P_6 in query set Q_2 . We start to

```

01 Function CheckConflict (COL: integer; Qr : set of pages;): boolean;
02 begin
03   CheckConflict := False;
04   i := 1;
05   while (i ≤ AC) and (CheckConflict = False) do
06     begin
07       if (M[i, COL] ∈ Qr) then CheckConflict := True;
08       i := i + 1;
09     end;
10 end;

```

Fig. 16. The *CheckConflict* function.

check slot 1 of channel 1, and we find that the page in $M[1, 1]$ is page P_3 , which is also in query set Q_2 . Therefore, slot 1 is a conflicting slot for page P_3 and function *CheckConflict* terminates with the value, *True*. For another example, when page P_7 in query set Q_3 is ready to be assigned to channel 3, we want to check whether slot 1 is a conflicting slot for page P_7 . We find that slot 1 in all channels does not contain any page in query set Q_3 , so function *CheckConflict* terminates with the value, *False*.

Data pages	P_1	P_2	P_3	P_4	P_5	P_8	P_6	P_{10}	P_{12}	P_9	P_7	P_{11}	P_{13}	P_{15}	P_{14}
Access frequency	92	68	55	55	35	32	25	25	25	20	8	8	7	7	5

Fig. 17. The data pages of Example 2 after sorting on their frequencies.

Query list	Data pages
TQ_1	P_1, P_2, P_3, P_4, P_5
TQ_2	$P_1, P_2, P_6, P_{10}, P_{12}$
TQ_3	P_1, P_3, P_4, P_8, P_9
TQ_4	P_2, P_7, P_{11}
TQ_5	P_1, P_8, P_{13}, P_{15}
TQ_6	P_1, P_8, P_{14}

Fig. 18. TQ_i in Example 2 ($1 \leq i \leq NQ$).

$$m = \begin{bmatrix} P_1 & P_2 & P_3 & P_4 & P_5 & P_{14} \\ P_7 & & P_6 & P_{10} & P_{12} & \\ & P_8 & P_{11} & P_{13} & P_9 & P_{15} \end{bmatrix}$$

Fig. 19. The output matrix of Example 2 based on the hybrid approach.

The purpose of function *TryAvailSlot* (ROW, Q_r) is to try to find a non-conflicting slot in list AS_{ROW} for a page P_k . If there is more than one non-conflicting slot, we prefer the one near the left end. (This is because we wish as far as possible to assign a page in a preceding slot.) Function *TryAvailSlot* terminates when we find a non-conflicting slot. (Note that function *TryAvailSlot* terminates even if some slots in the set AS_{ROW} have not

been checked.) On the other hand, if we cannot find a non-conflicting slot among all the elements in list AS_{ROW} , then function $TryAvailSlot$ terminates and returns -1. In function $TryAvailSlot$, we also call function $CheckConflict$ to check whether a slot is a conflicting slot for page P_k . For the previous example, when we want to assign page P_{11} to channel 2, we try to find a non-conflicting slot in list AS_2 . At this point, list AS_2 contains slot 1. Since slot 1 is not a conflicting slot for page P_{11} ($\in Q_4$) (i.e., $CheckConflict(1, Q_4) = False$), function $TryAvailSlot$ returns 1. For another example, when we want to assign page $P_{12} \in Q_2$ to channel 3, we try to find a non-conflicting slot in list AS_3 first. At this point, list AS_3 contains slot 2. Since slot 2 will cause page P_{12} to conflict with page P_4 (i.e., $CheckConflict(2, Q_5) = True$), function $TryAvailSlot$ returns -1.

Query set	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6
Access time	5	5	5	3	6	6

Fig. 20. The expected access time of each query set for Example 2 based on the hybrid approach.

Query set	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6
Access time	5	5	5	6	6	7

Fig. 21. The expected access time of each query set for Example 2 based on the SNV strategy.

We will use another example (Example 2) shown in Fig. 6 to show how the hybrid approach reduces the chance of bandwidth being wasted. The tracing table of each page after processing procedure $AssignPosition$ is shown in Table 1, and the result of the broadcast matrix is shown in Fig. 19. Note that after assigning page P_{15} to $M[3, 6]$, we have $SC = 6$ as shown in Table 1. This is because there is no non-conflicting slot in any of the three channels for page P_{15} , page P_{15} is assigned to the extended slot $((SC + 1) = 6)$ of channel 3. Now we have $NS_3 = 7$ and update $SC = 6$. We extend the 3×5 broadcast matrix to a 3×6 matrix. Therefore, page P_{14} can be considered for assignment to (1) slot 2 of channel 2, (2) slot 1 of channel 3, (3) slot 6 of channel 1 or (4) slot 6 of channel 3. The procedure steps for page P_{14} are shown in Table 2. In the first loop, we find that slot 1 of channel 3 will cause page P_{14} to conflict with page P_1 . In the next loop, after step A, we have $NS_1 (= 6) \leq SC (= 6)$; therefore, step D will be performed. In step D, we have $COL = NS_1 = 6$. Since slot 6 is not a conflicting page for page P_{14} , step E will be skipped. Moreover, we have $COL (= 6) \leq SC (= 6)$, so step G will be performed. In step G, page P_{14} is assigned to $M[ROW, COL]$, i.e., $M[1, 6]$. Finally, there are only 3 empty slots in the broadcast matrix generated by the hybrid approach, as compared to 6 empty slots in the broadcast matrix generated by the SNV strategy. Fig. 20 and Fig. 21 show the expected access time of each query set based on the hybrid approach and the SNV strategy, respectively. The total expected access time based on the hybrid approach is as follows:

$$\begin{aligned}
 TEAT &= \sum_{i=1}^6 R_i \times AT_i \\
 &= (5 \times 35 + 5 \times 25 + 5 \times 20 + 3 \times 8 + 6 \times 7 + 6 \times 5) / 100 = 4.96.
 \end{aligned}$$

On the other hand, the total expected access time based on the SNV strategy is as follows:

$$TEAT = \sum_{i=1}^6 R_i \times AT_i$$

$$= (5 \times 35 + 5 \times 25 + 5 \times 20 + 3 \times 8 + 6 \times 7 + 7 \times 5) / 100 = 5.01.$$

Table 1. A tracing result of each page of Example 2 after procedure *AssignPosition* is performed.

Query List (r)	Page	ROW	COL	u	NS ₁	NS ₂	NS ₃	AS ₁	AS ₂	AS ₃	SC	M
TQ ₁	-	1	-	-1	1	1	1	<>	<>	<>	5	-
TQ ₁	P ₁	1	1	-1	2	1	1	<>	<>	<>	5	M[1, 1]
TQ ₁	P ₂	1	2	-1	3	1	1	<>	<>	<>	5	M[1, 2]
TQ ₁	P ₃	1	3	-1	4	1	1	<>	<>	<>	5	M[1, 3]
TQ ₁	P ₄	1	4	-1	5	1	1	<>	<>	<>	5	M[1, 4]
TQ ₁	P ₅	1	5	-1	6	1	1	<>	<>	<>	5	M[1, 5]
TQ ₂	-	1	5	-1	6	1	1	<>	<>	<>	5	-
TQ ₂	P ₆	2	3	-1	6	4	1	<>	<>	<>	5	M[2, 3]
TQ ₂	P ₁₀	2	4	-1	6	5	1	<>	<1, 2>	<>	5	M[2, 4]
TQ ₂	P ₁₂	2	5	-1	6	6	1	<>	<1, 2>	<>	5	M[2, 5]
TQ ₃	-	3	5	-1	6	6	1	<>	<1, 2>	<>	5	-
TQ ₃	P ₈	3	2	-1	6	6	3	<>	<1, 2>	<1>	5	M[3, 1]
TQ ₃	P ₉	3	5	-1	6	6	6	<>	<1, 2>	<1, 3, 4>	5	M[3, 5]
TQ ₄	-	1	5	-1	6	6	6	<>	<2>	<1, 3, 4>	5	-
TQ ₄	P ₇	2	5	1	6	6	6	<>	<2>	<1, 3, 4>	5	M[2, 1]
TQ ₄	P ₁₁	3	5	3	6	6	6	<>	<2>	<1, 4>	5	M[3, 3]
TQ ₅	-	2	5	3	6	6	6	<>	<2>	<1, 4>	5	-
TQ ₅	P ₁₃	3	5	4	6	6	6	<>	<2>	<1>	5	M[3, 4]
TQ ₅	P ₁₅	3	6	-1	6	6	7	<>	<2>	<1>	6	M[3, 6]
TQ ₆	-	3	6	-1	6	6	7	<>	<2>	<1>	6	-
TQ ₆	P ₁₄	1	6	-1	7	6	7	<>	<2>	<1>	6	M[1, 6]

Table 2. A case of pages P₁₄ through steps ACADG.

		P ₁₄											
Loop	Step	C	u	ROW	COL	NS ₁	NS ₂	NS ₃	AS ₁	AS ₂	AS ₃	SC	F ₆
0	-	0	-	3	6	6	6	7	<>	<2>	<1>	6	0
1	A	0	-1	3	6	6	6	7	<>	<2>	<1>	6	0
1	C	1	-1	1	6	6	6	7	<>	<2>	<1>	6	0
2	A	1	-1	1	6	6	6	7	<>	<2>	<1>	6	0
2	D	1	-1	1	6	6	6	7	<>	<2>	<1>	6	0
2	G	1	-1	1	6	7	6	7	<>	<2>	<1>	6	0

Consequently, the total expected access time based on the hybrid approach is less than that based on the SNV strategy.

4. PERFORMANCE STUDY

In this section, we study the performance of our hybrid approach and make a comparison with Ke's SNV strategy. Our experiments were performed on a PentiumIII 733 MHz, 128 MB of main memory, running Windows ME.

4.1 The Simulation Model

We generated synthetic query sets to evaluate the performance of the SNV strategy and the hybrid approach over a large range of data characteristics. The parameters used to generate synthetic query sets are shown in Table 3 [4]. The length of a query set is determined by a Poisson distribution with mean μ equal to $|MeanQ|$. The size of a query set is between $|MinQ|$ and $|MaxQ|$. The query sets generated have to satisfy the following three conditions: (1) query sets often have common data pages; (2) $Q_i \neq Q_j$, which indicates that no two query sets are the same; (3) $\bigcup_{i=1}^{NQ} Q_i = D$, which indicates that all pages in the database will occur in at least one query set. Data pages in the first query set are chosen randomly from the set of pages. To model the phenomenon that query sets often have common data pages (i.e., conflict), some fraction of data pages in the subsequent query set C are chosen from the previously generated query set P . We use an exponentially distributed random variable with mean equal to the *correlation level* to decide this fraction F for each query set. Therefore, in query set C , $|P| \times F\%$ pages are chosen from the first $|P| \times F\%$ pages in query set P . To avoid generating the same query sets, the remaining data pages are picked at random in the following ways:

1. The remaining data pages are picked at random from those pages which have not occurred in any previously generated query set.
2. If all pages have occurred in previously generated query sets P , the remaining data pages are picked at random D . Then, we check whether query set C , which is being generated, is the same as any previously generated query set P . If query set C is the same as any previous generated query set P , the last element of query set C will be regenerated again at random. The last element of query set C will be regenerated again and again, until query set C is not the same as any previously generated query P .

The request ratio of each query set is selected from an exponential distribution with mean equal to 1. The request ratios are normalized such that the sum of all the request ratios equals 1. For example, suppose the number of query sets is 5; i.e., $NQ = 5$. According to the exponential distribution with mean equal to 1, the request ratios for these 5 query sets with ID equal to 1, 2, 3, 4 and 5 are 0.37, 0.14, 0.05, 0.02 and 0.01, respectively, and the sum of these request ratios of the 5 query sets is 0.59. Therefore, after normalization, the request ratios for those 5 query sets are 0.63 ($= 0.37 \div 0.59$), 0.24 ($= 0.14 \div 0.59$), 0.08 ($= 0.05 \div 0.59$), 0.03 ($= 0.02 \div 0.59$) and 0.02 ($= 0.01 \div 0.59$), respectively.

Table 3. Parameters.

D	Number of data pages in database
$ MeanQ $	Average size of query sets
$ MinQ $	The minimal size of query sets
$ MaxQ $	The maximal size of query sets
NQ	Number of query sets
$correlation\ level$	The parameter to decide the similar fraction for pages in the adjacent query set

4.2 Performance Analysis

The total number of slots (denoted as TS) in a broadcast cycle based on the SNV strategy can be computed as follows:

$$TS = NS_{AC} - 1.$$

Take Example 2 shown in Fig. 6 for example; the result of the broadcast matrix based on the SNV strategy is shown in Fig. 7, and its TS can be calculated as follows:

$$TS = NS_3 - 1 = 8 - 1 = 7.$$

The total number of slots in a broadcast cycle based on the hybrid approach can be computed as follows:

$$TS = SC = NS_{AC} - 1.$$

Take Example 2 as shown in Fig. 6 as an example; the result of the broadcast matrix based on the SNV strategy is shown in Fig. 17; and its TS can be calculated as follows:

$$TS = SC = 6 = NS_3 - 1 = 7 - 1.$$

Moreover, the methods used to calculate the total number of empty slots (denoted as TES) in a broadcast cycle based on the SNV strategy and the hybrid approach are the same. The total number of empty slots in a broadcast cycle can be computed as follows:

$$TES = TS \times AC - D.$$

Take Example 2 (shown in Fig. 6) as an example, we have $TES = 7 \times 3 - 15 = 6$ based on the SNV strategy. For the same example, based on the hybrid approach, we have

$$TES = 6 \times 3 - 15 = 3.$$

Furthermore, the methods used to calculate the total expected delay time (denoted as $TEAT$) based on the SNV strategy and the hybrid approach are the same. The total expected delay time can be computed as follows:

$$TEAT = \sum_{r=1}^{NQ} R_r \times AT_r.$$

AT_r can be calculated based on the following two policy.

Let SPN_r^i be the number of pages ($\in Q_r$) which are assigned to slot i and let $MaxSPN$ be the maximal value of SPN_r^i , $1 \leq i \leq (NS_{AC} - 1)$. If $MaxSPN = 1$, this indi-

cates that the whole query set can be retrieved in a broadcast cycle. Therefore, the last slot where the last page in a query set is broadcast is the total access time of the query set. However, if $MaxSPN$ is greater than 1, this indicates that the whole query set can not be retrieved in a broadcast cycle. Therefore, we can calculate AT_r based on the following policies:

1. If the maximal value of SPN_r^i (called $MaxSPN$) equals 1, $1 \leq i \leq (NS_{AC} - 1)$, then AT_r is the maximal value of i with $SPN_r^i = 1$.
2. If $MaxSPN > 1$, $1 \leq i \leq (NS_{AC} - 1)$, we let MP be the maximal value of i which has $SPN_r^i = MaxSPN$. Then, AT_r can be computed as follows:

$$AT_r = TS \times (MaxSPN - 1) + MP.$$

Take Example 2 shown in Fig. 6 as an example; based on the SNV strategy, the result of the broadcast matrix is shown in Fig. 7. Since pages P_1, P_8 and P_{14} in query set Q_6 are assigned to slots 1, 2 and 7, respectively, we have $SPN_6^1 = SPN_6^2 = SPN_6^7 = 1$ and $SPN_6^3 = SPN_6^4 = SPN_6^5 = SPN_6^6 = 0$. The maximal value of SPN_6^i is 1. Therefore, the total access time of query set Q_6 based on the SNV strategy, which is denoted as AT_6 , equals 7, which is the maximal value of 1, 2 and 7. For the same example, based on the hybrid approach, the result of the broadcast matrix is shown in Fig. 7. Since pages P_1, P_8 and P_{14} in query set Q_6 are assigned to slots 1, 2 and 6, respectively, we have $SPN_6^1 = SPN_6^2 = SPN_6^6 = 1$ and $SPN_6^3 = SPN_6^4 = SPN_6^5 = 0$. The maximal value of SPN_6^i is 1. Therefore, the total access time of query set Q_6 in the SNV strategy, which is denoted as AT_6 , equals 6, which is the maximal value of 1, 2 and 6.

4.3 Simulation Results

In this section, we study the performance of the hybrid approach and compare it with Ke's SNV strategy [22]. In the simulation, we let $D = 800$, $correlation = 0.5$ and $|MinQ| = 1$. We considered 27 test samples, which included the combinations $|MeanQ| = 7, 9$ and 11, $|MaxQ| = 2 \times |MeanQ|$, $NQ = 200, 300$ and 400, and $AC = 3, 5$ and 7. For each test sample, we computed the average result for 100 executions. The parameters and their default settings are shown in Table 4. To ensure that all the pages in the database have to occur at least in a query set, the values of the parameters in our simulation have to satisfy the following condition: $NQ \times |MeanQ| \geq D$.

Table 4. The parameters and their default settings used in the simulation of multiple channel.

Parameter	Default value
D	800
$ MeanQ $	7, 9, 11
$ MinQ $	1
$ MaxQ $	$2 \times MeanQ $
NQ	200, 300, 400
$correlation\ level$	0.5
AC	3, 5, 7

A broadcast cycle starts at the first slot and ends at the slot where the last page in the database is broadcast. Therefore, the total number of slots in a broadcast cycle, which is denoted as TS , can be calculated as $NS_{AC} - 1$ based on the SNV strategy and the hybrid approach. Since the SNV strategy only extends the number of slots in the last channel, NS_{AC} based on the SNV strategy is greater than it is based on the hybrid approach. Obviously, the total number of slots based on the SNV strategy is greater than that based on the hybrid approach. When $NQ = 200, 300$ and 400 , the detailed simulation results for the total number of slots in a broadcast cycle based on the hybrid approach and the SNV strategy for 100 executions are those shown in Tables 5, 6 and 7, respectively, where *Hybrid* denotes our proposed hybrid approach and *SNV* denotes Ke's SNV strategy. From the results, we find that our hybrid approach always generates a smaller number of slots than the SNV strategy does. As AC increases, the total number of slots decreases based on both the hybrid approach and the SNV strategy. As $|MeanQ|$ is increased, the total number of slots is increased in both the hybrid approach and the SNV strategy.

Table 5. A comparison of total number of slots in a broadcast cycle ($D = 800, NQ = 200$).

$ MeanQ $	<i>Hybrid</i>			<i>SNV</i>		
	$AC = 3$	$AC = 5$	$AC = 7$	$AC = 3$	$AC = 5$	$AC = 7$
7	268.51	163.77	117.59	268.54	165.04	117.90
9	269.06	164.66	118.66	269.25	166.09	119.18
11	269.76	165.18	120.18	269.94	166.80	121.38

Table 6. A comparison of total number of slots in a broadcast cycle ($D = 800, NQ = 300$).

$ MeanQ $	<i>Hybrid</i>			<i>SNV</i>		
	$AC = 3$	$AC = 5$	$AC = 7$	$AC = 3$	$AC = 5$	$AC = 7$
7	268.57	163.85	117.37	268.57	164.93	117.58
9	269.02	164.46	118.48	269.05	166.08	119.27
11	269.72	165.11	120.10	270.08	167.09	121.30

Table 7. A comparison of total number of slots in a broadcast cycle ($D = 800, NQ = 400$).

$ MeanQ $	<i>Hybrid</i>			<i>SNV</i>		
	$AC = 3$	$AC = 5$	$AC = 7$	$AC = 3$	$AC = 5$	$AC = 7$
7	268.60	163.71	117.72	268.66	164.88	117.94
9	269.08	164.49	118.53	269.27	165.99	119.10
11	269.62	165.45	119.63	269.98	167.17	121.16

Table 8. A comparison of total number of empty slots in a broadcast cycle ($D = 800, NQ = 200$).

$ MeanQ $	<i>Hybrid</i>			<i>SNV</i>		
	$AC = 3$	$AC = 5$	$AC = 7$	$AC = 3$	$AC = 5$	$AC = 7$
7	5.53	18.85	23.13	5.62	25.20	25.30
9	7.18	23.30	30.62	7.75	30.45	34.26
11	9.28	25.90	41.26	9.82	34.00	49.66

Table 9. A comparison of total number of empty slots in a broadcast cycle ($D = 800$, $NQ = 300$).

$ MeanQ $	Hybrid			SNV		
	$AC = 3$	$AC = 5$	$AC = 7$	$AC = 3$	$AC = 5$	$AC = 7$
7	5.71	19.25	21.59	5.71	24.65	23.06
9	7.06	22.30	29.36	7.15	30.40	34.89
11	9.16	25.55	40.70	10.24	35.45	49.10

The total number of empty slots in a broadcast cycle equals $(TS \times AC - D)$. Since TS based on the SNV strategy is usually greater than it is based on the hybrid approach, obviously, the total number of empty slots in a broadcast cycle based on the hybrid approach is greater than that based on the SNV strategy. When $NQ = 200, 300$ and 400 , the detailed simulation results for the total number of empty slots in a broadcast cycle based on the hybrid approach and the SNV strategy for 100 executions are those shown in Tables 8, 9 and 10, respectively. From the results, we find that our hybrid approach always generates a smaller number of empty slots than the SNV strategy does. As AC increases, the total number of empty slots increases based on both the hybrid approach and the SNV strategy. As $|MeanQ|$ increases, the total number of empty slots increases based on both the hybrid approach and the SNV strategy.

Table 10. A comparison of total number of empty slots in a broadcast cycle ($D = 800$, $NQ = 200$).

$ MeanQ $	Hybrid			SNV		
	$AC = 3$	$AC = 5$	$AC = 7$	$AC = 3$	$AC = 5$	$AC = 7$
7	5.80	18.55	24.04	5.98	24.40	25.58
9	7.24	22.45	29.71	7.81	29.95	33.70
11	8.86	27.25	37.41	9.94	35.85	48.12

Table 11. A comparison of total number of expected access in a broadcast cycle ($D = 800$, $NQ = 200$).

$ MeanQ $	Hybrid			SNV		
	$AC = 3$	$AC = 5$	$AC = 7$	$AC = 3$	$AC = 5$	$AC = 7$
7	43.167	27.822	21.320	43.990	28.745	22.310
9	53.406	34.309	26.831	54.413	35.407	28.023
11	64.169	41.493	32.065	65.382	42.876	33.414

Table 12. A comparison of total number of expected access in a broadcast cycle ($D = 800$, $NQ = 300$).

$ MeanQ $	Hybrid			SNV		
	$AC = 3$	$AC = 5$	$AC = 7$	$AC = 3$	$AC = 5$	$AC = 7$
7	43.245	27.822	21.278	44.134	28.782	22.341
9	53.377	34.535	26.751	54.419	35.630	27.940
11	64.702	41.695	31.862	65.848	42.897	33.246

Table 13. A comparison of total number of expected access in a broadcast cycle ($D = 800$, $NQ = 400$).

MeanQ	Hybrid			SNV		
	AC = 3	AC = 5	AC = 7	AC = 3	AC = 5	AC = 7
7	42.762	27.841	21.384	43.638	28.794	22.381
9	53.280	34.678	26.530	54.279	35.828	27.605
11	64.591	41.573	31.847	65.830	42.849	33.217

When $NQ = 200, 300$ and 400 , the detailed simulation results for the total expected access time based on the hybrid approach and the *SNV* strategy for 100 executions are those shown in Tables 11, 12 and 13, respectively. From the results, we find that our hybrid approach always requires a shorter expected access time than the *SNV* strategy does. As *AC* increases, the total expected access time decreases based on both the hybrid approach and the *SNV* strategy. As $|MeanQ|$ increases, the total expected access time increases based on both the hybrid approach and the *SNV* strategy.

When $NQ = 200, 300$ and 400 , the detailed simulation results for the average expected access time of each query set based on the hybrid approach and the *SNV* strategy for 100 executions are those shown in Tables 14, 15 and 16, respectively. From the results, we find that our hybrid approach always requires a shorter expected access time than the *SNV* strategy does. As *AC* increases, the average expected access time of each query set decreases based on both the hybrid approach and the *SNV* strategy. As $|MeanQ|$ increases, the average expected access time of each query set increases based on both the hybrid approach and the *SNV* strategy.

Table 14. A comparison of average expected access time of each query set ($D = 800$, $NQ = 200$).

MeanQ	Hybrid			SNV		
	AC = 3	AC = 5	AC = 7	AC = 3	AC = 5	AC = 7
7	169.39	104.30	76.79	170.24	105.58	77.67
9	189.76	118.98	89.14	191.03	120.28	90.16
11	207.11	132.63	100.00	208.87	134.13	101.63

Table 15. A comparison of total expected access in a broadcast cycle ($D = 800$, $NQ = 300$).

MeanQ	Hybrid			SNV		
	AC = 3	AC = 5	AC = 7	AC = 3	AC = 5	AC = 7
7	192.93	119.55	88.45	193.24	120.57	89.38
9	210.70	133.64	100.13	211.83	134.24	100.81
11	227.36	146.54	110.88	227.74	147.34	111.68

Table 16. A comparison of average expected access in a broadcast cycle ($D = 800$, $NQ = 400$).

MeanQ	Hybrid			SNV		
	AC = 3	AC = 5	AC = 7	AC = 3	AC = 5	AC = 7
7	204.30	127.40	94.35	205.12	127.91	95.06
9	221.98	140.81	105.87	222.64	141.49	106.15
11	237.82	153.14	116.55	238.25	153.76	117.24

5. CONCLUSIONS

In many situations, a mobile client might need data from more than one page. Moreover, when a number of wireless channels are available, designing scheduling strategies becomes more difficult. In this paper, we have proposed an efficient hybrid approach to query set broadcast scheduling for multiple channels. Through our performance analysis and simulation, we have shown that our hybrid approach requires a shorter total expected delay access time and creates a smaller number of total slots and a smaller number of empty slots in one broadcast cycle than the *SNV* strategy does. In an asymmetric environment, when a client receives an information item containing errors (due to some environmental disturbance), it is not always possible for the client to request retransmission of the information [32]. In this case, the client must wait for the next transmission of the required item. In real-time environments, minimizing the average latency time is no longer the main criteria for performance [8]. Rather, guaranteeing that time constraints are met is the most important concern. Therefore, developing ways to efficiently design broadcast programs in real-time environments is possible future research direction.

REFERENCES

1. S. Acharya, M. Franklin, S. Zdonik, and R. Alonso, "Broadcast disks: data management for asymmetric communications environments," in *Proceedings of ACM SIGMOD International Conference on Management of Data*, 1995, pp. 199-210.
2. S. Acharya, M. Franklin, and S. Zdonik, "Dissemination-based data delivery using broadcast disks," *Personal Communications*, Vol. 2, 1995, pp. 50-60.
3. S. Acharya, M. Franklin, and S. Zdonik, "Prefetching from a broadcast disk," in *Proceedings of the 12th IEEE International Conference on Data Engineering*, 1996, pp. 276-285.
4. R. Agrawal and R. Srikant, "Fast algorithm for mining association rules in large databases," in *Proceedings of the 20th International Conference Very Large Data Base*, 1994, pp. 490-501.
5. M. H. Ammer and J. W. Wong, "The design of teletext broadcast cycles," *Performance Evaluation*, Vol. 5, 1985, pp. 235-242.
6. M. H. Ammar and J. W. Wong, "On the optimality of cyclic transmission in teletext systems," *IEEE Transactions on Communications*, Vol. 35, 1987, pp. 68-73.
7. D. Barbara and T. Imielinski, "Sleepers and workaholics: caching strategies for mobile environments," in *Proceedings of ACM SIGMOD Conference on Management of Data*, 1994, pp. 1-12.
8. D. Barbara, "Mobile computing and database-A survey," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 11, 1999, pp. 108-117.
9. S. Baruah and A. Bestavros, "Pinwheel scheduling for fault-tolerant broadcast disks in real-time database systems," in *Proceedings of the 13th IEEE International Conference on Data Engineering*, 1997, pp. 543-551.
10. A. Bestavros, "AIDA-based real-time fault-tolerant broadcast disks," in *Proceedings of the Real-Time Technology and Applications Symposium*, 1996, pp. 49-58.
11. T. F. Bowen, G. Gopal, G. Herman, T. Hickey, K. C. Lee, W. H. Mansfield, J. Raitz, and A. Weinrib, "The datacycle architecture," *Communications of ACM*, Vol. 35,

- 1992, pp. 71-81.
12. Y. D. Chung and M. H. Kim, "QEM: a scheduling method for wireless broadcast data," in *Proceedings of the 6th International Conference on Database Systems for Advanced Applications*, 1999, pp. 135-142.
 13. M. H. Dunham and A. Helal, "Mobile computing and databases: anything new?" *ACM SIGMOD Record*, Vol. 24, 1995, pp. 5-9.
 14. V. Gondhalekar, R. Jain, and J. Werth, "Scheduling on airdisks: efficient access to personalized information services via periodic wireless data broadcast," in *Proceedings of 1997 International Conference on Communications Towards the Knowledge Millennium*, Vol. 3, 1997, pp. 1276-1280.
 15. S. Hameed and N. Vaidya, "Efficient algorithm for scheduling data broadcast," *Wireless Networks*, Vol. 5, 1999, pp. 183-193.
 16. G. Herman, G. Gopal, K. Lee, and A. Weinrib, "The datacycle architecture for very high throughput database systems," in *Proceedings of ACM SIGMOD International Conference on Management of Data*, 1987, pp. 97-103.
 17. T. Imielinski and B. Badrinath, "Mobile wireless computing: challenges in data management," *Communications of ACM*, Vol. 37, 1994, pp. 18-28.
 18. T. Imielinski and H. Korth, *Mobile Computing*, Kluwer Academic Publishers, Chapter 12, 1996, pp. 331-361.
 19. T. Imielinski, S. Viswanathan, and B. R. Badrinath, "Power efficient filtering of data on air," in *Proceedings of the Fourth International Conference on Extending Database Technology*, 1994, pp. 245-258.
 20. T. Imielinski, S. Viswanathan, and B. R. Badrinath, "Energy efficient indexing on air," in *Proceedings of ACM SIGMOD International Conference on Management of Data*, 1994, pp. 25-36.
 21. T. Imielinski, S. Viswanathan, and B. R. Badrinath, "Data on air: organization and access," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 9, 1997, pp. 353-371.
 22. C. H. Ke, "Broadcast scheduling for multiple channels in wireless information systems," in *Proceedings of 1999 National Computer Symposium*, Vol. 3, 1999, pp. 525-532.
 23. C. Kenyon, N. Schabanel, and N. Young, "Polynomial-time approximation scheme for data broadcast," in *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, 2000, pp. 659-666.
 24. D. L. Lee and W. C. Lee, "Using signature techniques for information filtering in wireless and mobile environments," *Special Issue on Databases and Mobile Computing, Journal on Distributed and Parallel Databases*, Vol. 4, 1996, pp. 205-227.
 25. V. C. S. Lee, K. W. Lam, S. Wu, and E. Chan, "Broadcasting consistent data in mobile computing environments," in *Proceedings of the 7th IEEE Symposium on Real-Time Technology and Application*, 2001, pp. 123-124.
 26. W. C. Peng and M. S. Chen, "Dynamic generation of data broadcasting programs for a broadcast disk array in a mobile computing environment," in *Proceedings of the 9th International Conference on Information Knowledge Management*, 2000, pp. 38-45.
 27. N. Schabanel, "The data broadcast problem with preemption," in *Proceedings of the 17th International Symposium on Theoretical Aspects of Computer Science*, 2000, pp. 181-192.
 28. K. L. Tan and J. X. Yu, "A dynamic schedule for the infinite air-cache," *IEEE*

- Transactions on Knowledge and Data Engineering*, Vol. 24, 1997, pp. 97-112.
29. K. L. Tan and B. C. Ooi, "On selective tuning in unreliable wireless channels," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 28, 1998, pp. 209-231.
 30. K. L. Tan and L. X. Yu, "Generating broadcast programs that support range queries," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 10, 1998, pp. 668-672.
 31. K. L. Tan, J. X. Yu, and P. K. Enk, "Supporting range queries in a wireless environment with nonuniform broadcast," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 29, 1999, pp. 201-221.
 32. N. H. Vaidya and S. Hameed, "Scheduling data broadcast in asymmetric communication environments," *Wireless Networks*, Vol. 5, 1999, pp. 171-182.
 33. K. L. Wu, P. S. Yu, and M. S. Chen, "Energy-efficient caching for wireless mobile computing," in *Proceedings of the 12th International Conference on Data Engineering*, 1996, pp. 336-345.
 34. P. Xuan, S. Sen, O. Gonzalez, J. Fernandez, and K. Ramamritham, "Broadcast on demand: efficient and timely dissemination of data in mobile environments," in *Proceedings of 1997 Real-Time Technology and Applications Symposium*, 1997, pp. 38-48.
 35. S. Zdonik, M. Franklin, R. Alonso, and S. Acharya, "Are 'Disks in Air' just pie in the sky?" *IEEE Workshop on Mobile Computing Systems and Applications*, 1994, pp. 12-19.



Ye-In Chang (張玉盈) was born in Taipei, Taiwan, R.O.C., in 1964. She received the B.S. degree in computer science and information engineering from National Taiwan University, Taipei, Taiwan, in 1986, and M.S. and Ph.D. degrees in computer and information science from the Ohio State University, Columbus, Ohio, in 1987 and 1991, respectively.

From August 1991 to July 1999, she joined the faculty of the department of applied mathematics at National Sun Yat-Sen University, Kaohsiung, Taiwan. Since August 1997, she has been a Professor in the department of applied mathematics at National Sun Yat-Sen University, Kaohsiung, Taiwan. Since August 1999, she has been a Professor in the department of computer science and engineering at National Sun Yat-Sen University, Kaohsiung, Taiwan. Her research interests include database systems, distributed systems, multimedia information systems and mobile information systems.



Shih-Ying Chiu (邱詩穎) was born in Taipei, Taiwan, R.O.C., in 1977. She received the B.S. degree in Applied Mathematic in 1999 and M.S. degree in Computer Science and Engineering in 2001 both from National Sun Yat-Sen University. She is current a software process engineer of Trend Micro corporation in Taiwan. She is doing some developing jobs about software configuration management.