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

Ye-In Chang, and Shih-Ying Chiu

Dept. of Computer Science and Engineering
National Sun Yat-Sen University
Kaohsiung, Taiwan
Republic of China
{E-mail: changyi@cse.nsysu.edu.tw}
{Tel: 886-7-5252000 (ext. 4334)}
{Fax: 886-7-5254301}

Abstract

In this paper, we concern about the broadcasting scheduling over multiple channels under the case that a mobile client might need data of 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 sets broadcast scheduling in multiple channels. In the SNV strategy, the data pages of the same query set are put as together 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 in 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 results in the long access time for requiring the whole query set; (2) it may extend the number of slots in a certain chain, which results in the 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. From our performance analysis and simulation, we show that our hybrid approach requires shorter total expected delay access time, and creates smaller number of slots and smaller number of empty slots in one broadcast cycle than the SNV strategy.

(Key Words: access time, bandwidth, broadcast schedule, mobile information systems, multiple channels.)

\footnote{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.}
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, the design of 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 the communications, frequent disconnections, 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 asymmetric, along with the restriction in power that the mobile units have, make the model of broadcasting data to the clients, 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 a uplink channel and servers respond by sending data to clients on a downlink channel.

Although the on-demand mode lets mobile clients can play more active role in getting data they need, it is not suitable in asymmetric communication environment on account of upstream narrower communication capabilities. This restriction makes only the limited clients be served in the on-demand mode. On the other hand, under the broadcasting mode, the number of mobile clients which can simultaneously listen to the downlink channel could be almost infinite scaled. In wireless computing environment, to use the broadcasting mode not only can save the bandwidth of the uplink channel and but also can be scale to any number of mobile clients who listen to the publishing 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 that is equal to the time to broadcast a page. The amount of time a client has to wait for an information item that it needs is called access time. The access time depends on the broadcast schedule. A good broadcast program can minimize the mean access time.

There have been many strategies proposed for efficient broadcast delivery [5, 6, 11, 13, 15, 16, 23, 25, 26, 27]. Basically, those 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, there are some researches on reducing access time by caching [2, 3, 7, 18, 33], nonuniform broadcast [3, 18, 30, 31], and those on reducing tuning time such as indexing [14, 19, 20, 21, 29, 33, 35], hashing [19] and signature [24] techniques, where tuning time means the among of time spent by a client listening to the channel. Strategies presented in [9, 10], considered real-time, fault-tolerant, secure broadcast organization technique. Strategies presented in [30, 31] generated 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 of more than one page. For example, users often need prices of one or more stocks and the traffic information of a few roads. Under these circumstances, how to minimize the access time for the 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 the broadcast stream is illustrated in Figure 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 Figure 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 \( <d_2, d_3, d_1, d_4, d_5> \), then the client can retrieve \( d_1, d_4 \) in current bcast. This says that an efficient schedule provides access time reduction to mobile
Figure 1: A query contains a set of data objects

Moreover, all of the past works discuss how to broadcast the data over single channel. In [22], Ke studied the issue of scheduling the broadcast data on multiple wireless channels as an MC may access more than one data page, and has proposed the page-based scheduling strategies and the set-based scheduling strategies. In the paged-based scheduling strategies, the PHV strategy and the PVV strategy, a data page with the maximal access frequency is broadcast first. Therefore, the clients can retrieve the desired data pages as fast as possible [22]. However, 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, it will result in an increase of the total access time. In the set-based strategies, the $SHV$ strategy and the $SVV$ strategy, the data pages of the same query set are put as together as possible. Moreover, the data pages of the same query set are assigned to different time slots to avoid conflicts. However, both of the $SHV$ strategy and the $SVV$ strategy still can not guarantee no conflict between data pages to occur. In order to reduce the probability of conflicts between data pages, Ke has 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 predictive slots in the broadcast scheduling matrix cause a conflict for
a certain page, the page will be assigned to the extended slot of the last channel, where the value of predictive slots is equal to [the number of data pages / the number of channels]. That will result in a waste of bandwidth.

Therefore, we propose an efficient broadcast scheduling strategy, a hybrid approach to improve these two above disadvantages. Basically, the hybrid approach combines the advantages of the page-based and the set-based strategies. 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 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 the broadcast channels from top to down, while pages in a query set are assigned to the broadcast channels from right to left.

The rest of 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 SNV strategy. In section 3, we present our hybrid approach. In section 4, we study the performance of our hybrid approach, and make a comparison with Ke’s SNV strategy. Finally, section 5 gives the conclusion.

### 2 Background

In this section, we first describe Ke’s SNV strategy. Next, we give two examples to show the disadvantages of Ke’s SNV strategy.
m = \begin{bmatrix}
P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 & P_9 & P_{10} & P_{11} & P_{12} & P_{13} & P_{14} & P_{15} 
\end{bmatrix}

Figure 3: The output matrix of Example 1 based on the SNV strategy

2.1 The SNV Strategy

The basic idea of the SNV strategy is to arrange all data sets to the available channels in a round-robin pattern. 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 Figure 2, Figure 3 shows the resulting broadcast matrix (M) in the SNV strategy and Figure 4 shows the expected access time of each set.

Now, we discuss how the data pages of query set Q_4 are scheduled. (Note that before pages in query set Q_4 are scheduled, pages P_1, P_2, P_3, P_4, P_5, P_6, P_7 and P_8 have been 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 (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 a 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 slot 2 of channel 2. Up to this point, pages in query set Q_4 have been all 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, 6 of channel 3, respectively. (Note that slot 3 of channel 3 can not be assigned 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) with the expected delay for that set (AT_i) and summing the results. Therefore, the expected access time of this example is as follows:

\[
\sum_{i=1}^{N_Q} R_i \times AT_i = \sum_{i=1}^{6} R_i \times AT_i = (4 \times 35 + 3 \times 25 + 4 \times 20 + 5 \times 8 + 5 \times 7 + 6 \times 5)/100 = 4.
\]
Query Sets Broadcasting Scheduling

<table>
<thead>
<tr>
<th>Query set</th>
<th>$Q_1$</th>
<th>$Q_2$</th>
<th>$Q_3$</th>
<th>$Q_4$</th>
<th>$Q_5$</th>
<th>$Q_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access time</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

Figure 4: The expected access time of each query set for Example 1 based on the SNV strategy

<table>
<thead>
<tr>
<th>Data pages</th>
<th>$P_3$</th>
<th>$P_4$</th>
<th>$P_2$</th>
<th>$P_1$</th>
<th>$P_6$</th>
<th>$P_5$</th>
<th>$P_{14}$</th>
<th>$P_{13}$</th>
<th>$P_8$</th>
<th>$P_{12}$</th>
<th>$P_7$</th>
<th>$P_9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access frequency</td>
<td>60</td>
<td>55</td>
<td>43</td>
<td>35</td>
<td>32</td>
<td>20</td>
<td>20</td>
<td>15</td>
<td>13</td>
<td>8</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>

Figure 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 request ratios of them as shown in Figure 2, Figure 3 shows the result based on the SNV strategy. Note that based on the access frequency of each page as shown in Figure 5, page $P_3$ has the highest access frequency; however, it is assigned to slot 3 of channel 1, instead of 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 a random order, so a data page with high access frequency may be scheduled at the time slot near the end of the broadcast cycle. That will result in an increase of the total access time.

Let’s see another input example as shown in Figure 6. 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 a channel is $[15/3] = 5$. Figure 7 shows the result based on the SNV strategy. When page $P_{15}$ in query set $Q_5$ is prepared to be assigned to the broadcast matrix, there are two candidate slots in the broadcast matrix: (1) slot 2 of channel 2, and (2) slot 1 of channel 3. However, slot 2 of channel 2 will cause page $P_{15}$ conflicting with page $P_8$, and slot 1 of channel 3 will cause page $P_{15}$ conflicting with page $P_1$. Therefore, page $P_{15}$ has to be assigned to slot 6 of channel 3. For the same reason, page $P_{14}$ in query set $Q_6$ is assigned to slot 7 of channel 3. Finally, there are 6 empty slots in the broadcast matrix as shown.
<table>
<thead>
<tr>
<th>Query set</th>
<th>Data pages</th>
<th>Request ratio (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q_1$</td>
<td>$P_1, P_2, P_3, P_4, P_5$</td>
<td>35</td>
</tr>
<tr>
<td>$Q_2$</td>
<td>$P_1, P_2, P_6, P_{10}, P_{12}$</td>
<td>25</td>
</tr>
<tr>
<td>$Q_3$</td>
<td>$P_1, P_3, P_4, P_8, P_9$</td>
<td>20</td>
</tr>
<tr>
<td>$Q_4$</td>
<td>$P_2, P_7, P_{11}$</td>
<td>8</td>
</tr>
<tr>
<td>$Q_5$</td>
<td>$P_1, P_8, P_{13}, P_{15}$</td>
<td>7</td>
</tr>
<tr>
<td>$Q_6$</td>
<td>$P_1, P_8, P_{14}$</td>
<td>5</td>
</tr>
</tbody>
</table>

Figure 6: Example 2

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

Figure 7: The output matrix of Example 2 based on the SNV strategy

in Figure 7. However, although page $P_{14}$ does not conflict with page $P_{15}$, it is assigned to slot 7 of channel 3, instead of slot 6 of channel 1 or slot 6 of channel 2. Note that when all predictive slots in the broadcast scheduling matrix conflict for a certain page $P_k$, the SNV strategy always assigns page $P_k$ to the extended slot of the last channel. That will result in a waste of bandwidth.

3 The Hybrid Approach

The proposed hybrid approach combines the advantages of the page-based and the set-based strategies and also improves the two disadvantages of the SNV strategy.

3.1 Assumptions

This paper focuses on wireless broadcast environment. Some assumptions should be restricted in order to make our work feasible. These assumptions include: (1) The client population and their access patterns do not change. (2) Data is read-only. (3) Clients make no use of their upstream communications capability and retrieve required data items from the broadcast. (4) Clients are simple and without a great amount of memory. (5) When a client switches to the public channel, it can retrieve data pages immediately. (6)
The broadcast infrastructure is reliable. (7) The length of each page is fixed. (8) A query may contain more than one page. (9) There is no precedence relation in accessing data objects. (10) The server broadcasts pages over more than one channel. (11) Each data page is broadcast once within each broadcast cycle. (12) At each time slot, an MC can only tune into one channel to retrieve one data page. (13) The index is broadcast at the beginning of each broadcast cycle. Note that an index serves the purpose of allowing the MCs selectively tune into the channels to access the desired data pages, when they are broadcast. To avoid involving the detail of indexing techniques, the time of accessing index pages is ignored from the access time of MCs [22].

3.2 Variables

In the proposed algorithm, the following variables are used:

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 it indicates that page $P_i$ has not been assigned in the schedule; otherwise, it indicates that 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 the average; its initial value is computed as $SC = \lceil \frac{R_i}{AC} \rceil$;
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.

For the same input example as shown in Figure 2, Figure 8 shows the result based on the hybrid approach. Given $NQ = 6$, $D = 15$, $AC = 3$, we have $SC = \lceil D/AC \rceil = 5$. In our approach, first, we calculate the access frequency of each of these 15 pages and the result is shown in Figure 5. Next, we sort the pages ($P_i$) in each query set ($Q_i$) on their access frequency ($AF_i$) in a decreasing order, and assign the result to a temporary query list ($TQ_i$) as shown in Figure 9. Then, we assign each query list $TQ_i$ to the broadcast channels in a round-robin method. 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
\[ m = \begin{bmatrix} P_3 & P_4 & P_2 & P_1 & P_{14} \\ P_{11} & P_6 & P_{10} & P_9 & P_1 \\ P_7 & P_5 & P_8 & P_{12} & P_{13} \\ P_1 & P_{15} & P_2 & P_3 & P_{14} \end{bmatrix} \]

Figure 8: The output matrix of Example 1 based on the hybrid approach

<table>
<thead>
<tr>
<th>Query list</th>
<th>Data pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>TQ_1</td>
<td>P_3, P_4, P_2, P_1</td>
</tr>
<tr>
<td>TQ_2</td>
<td>P_3^*, P_6</td>
</tr>
<tr>
<td>TQ_3</td>
<td>P_1^*, P_7, P_8</td>
</tr>
<tr>
<td>TQ_4</td>
<td>P_2^*, P_{14}, P_{11}, P_9</td>
</tr>
<tr>
<td>TQ_5</td>
<td>P_6^*, P_{14}, P_{10}, P_{12}</td>
</tr>
<tr>
<td>TQ_6</td>
<td>P_{11}^*, P_5, P_{13}, P_{15}</td>
</tr>
</tbody>
</table>

\( P_i^* \): the repeating occurrence of page \( P_i \).

Figure 9: TQ_i in Example 1 (1 \( \leq \) i \( \leq \) NQ)

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 from slot 1 to slot 4 of channel 1, respectively. The next available slot of channel 1 is slot 5, i.e., \( NS_1 = 5 \). Up to 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 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 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 the slot is a conflicting slot for page \( P_k \). For the previous example, after assigning query list \( TQ_1 \) to the broadcast matrix, slot 1 is a conflicting slot for page \( P_5 \) 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 \( TQ_3 \),
Query Sets Broadcasting Scheduling

page $P_4$ is ignored due to 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 $N_S_3 = 4$ and list $A_S_3 = \langle 2 \rangle$.

Up to this point, 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 due to the same reason as page $P_3$.) Since $N_S_1 = 5$, page $P_{14}$ in query list $TQ_4$ is assigned to slot 5 of channel 1 and $N_S_1$ is updated to 6. After that, pages $P_{11}$ and $P_9$ in query list $TQ_4$ have not been assigned to the broadcast matrix, and slot 6 ($= N_S_1$) is out of the range of the slots of channel 1 ($N_S_1 (= 6) > SC (= 5)$). Therefore, we go down to the next channel, i.e., channel 2. At this time, since list $A_S_2$ is not empty, we will try the slot(s) in list $A_S_2$ first, instead of slot $N_S_2$. Therefore, although $N_S_2 = 3$, page $P_{11}$ is assigned to slot 1, instead of slot 3, of channel 2. That is because slot 1 recorded in list $A_S_2$ 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 a good slot for a page: (1) if $|A_S_{ROW}| \neq 0$, we try to find a non-conflicting slot in $A_S_{ROW}$; (2) if there is no non-conflicting slot in $A_S_{ROW}$, we try to find a non-conflicting slot from slot $N_S_{ROW}$; (3) if we can not find a non-conflicting slot based on the above two policies, we go down to the next channel. For page $P_9$ in query list $TQ_4$, it is assigned to slot 4, instead of slot 3, of channel 2, to avoid conflicting with page $P_2$ (in the same query set $Q_4$). Finally, we update $N_S_2$ as $N_S_2 = 5$.

When query list $TQ_5$ is considered, pages $P_6$ and $P_4$ are ignored due to 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 ($\in A_S_2$) of channel 2 as the same reason of page $P_{11}$ in query list $TQ_4$, and we update $A_S_2 = A_S_2 - 3$. For page $P_{12}$, slot 5 ($= N_S_2$) in channel 2 will cause page $P_{15}$ to conflict with page $P_{14}$, so we record slot 5 in list $A_S_2$ and let $N_S_2 = 6$. However, slot 6 ($= N_S_2$) 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 $A_S_3$ 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 ($= 4$).
$NS_3), instead of 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 due to the same reason as page $P_3$. Therefore, only pages $P_5$, $P_{13}$ and $P_{15}$ are considered. Moreover, since 6 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 due to 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 ($AS_1 = <>$ 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. Up to this point, 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 all three channels for page $P_{15}$, we could 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 $3 \times 6$. Figure 8 shows the result based on the hybrid approach. Figure 10 and Figure 11 show the expected access time of each query set in 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 ($R_i$) with the access time for that data set ($AT_i$) and summing the results. That is, $TEAT = \sum_{i=1}^{N} R_i \times AT_i$.

The total expected access time based on the hybrid approach is as follows:
\[
TEAT = \sum_{i=1}^{6} R_i \times AT_i \\
= (4 \times 35 + 2 \times 25 + 3 \times 20 + 5 \times 8 + 5 \times 7 + 6 \times 5)/100 = 3.55.
\]
The total expected access time based on the SNV strategy is as follows:
\[
TEAT = \sum_{i=1}^{6} R_i \times AT_i \\
= (4 \times 35 + 3 \times 25 + 4 \times 20 + 5 \times 8 + 5 \times 7 + 6 \times 5)/100 = 4.
\]
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 \(Q_1, Q_2, ..., Q_{NQ}\) with request ratios \(R_1, R_2, ..., R_{NQ}\), respectively, we have to assign all pages to the broadcast matrix, which is a \(AC \times SC\) matrix. The complete algorithm as shown in Figure 12 contains the following steps:

1. Call procedure \(HybridInitialization\) (as shown in Figure 13) to initialize the values of variables. In the \(HybridInitialization\) procedure, first, we calculate the number of available slots per channel on the average, which is denoted as \(SC\). Second, order the query sets \(Q_i\) from the hottest one to the coldest one based on their request ratios. Third, compute the access frequency \(AF_i\) for each page \(P_i\) and set \(F_i\) as 0, which indicates that all pages have not 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, 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 \(TQ_r\);
   \(TQ_r := TQ_r - < P_k >;\)
   if the page \(P_k\) has not been assigned to the broadcast matrix then

13
\textbf{Input:} The number of available channels, which is denoted as $AC$, and $NQ$ data
sets $Q_1, Q_2, \ldots, Q_{NQ}$ with request ratios $R_1, R_2, \ldots, R_{NQ}$, respectively;
\textbf{Output:} $M[i,j]$, a broadcast matrix, $1 \leq i \leq AC$, $1 \leq j \leq SC$;
begin
\quad Call \textit{HybridInitialization};
\quad For $r := 1$ to $NQ$ do
\quad \quad begin
\quad \quad \quad While ($|TQ_r| \neq 0$) do
\quad \quad \quad \quad begin
\quad \quad \quad \quad \quad Select the first page $P_k$ from $TQ_r$;
\quad \quad \quad \quad \quad $C := 0$;
\quad \quad \quad \quad \quad $TQ_r := TQ_r - < P_k >$
\quad \quad \quad \quad \quad while ($F_k = 0$) do
\quad \quad \quad \quad \quad \quad call \textit{AssignPosition}($P_k, Q_r, ROW$);
\quad \quad \quad \quad \quadend;
\quad \quad \quad If ($(r + 1) \mod AC) = 0$ then $ROW := 3$
\quad \quad \quad else $ROW := ((r + 1) \mod AC)$;
\quad \quad \quadend;
\quad \quad end;
\quad end.
\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{algorithm}
\caption{The \textit{Hybrid} algorithm}
\end{figure}

call procedure \textit{AssignPosition} (as shown in Figure 14) \par
to assign it to the broadcast matrix;
\end{verbatim}
end;
end.

In Figure 12, line 4 is what we do in step 1. From lines 5 to 17 is what we do in step 2. In step 2, we assign all input query sets to the broadcast matrix in a round-robin pattern. \par(Line 15 and line 16 are used for the process of the round-robin pattern.) The query set with the highest request ratio is assigned to the broadcast matrix first. Since we sort query sets 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 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

14
query list, if the page has not been assigned to the broadcast matrix, we call procedure 
\textit{AssignPosition} as shown in Figure 14 to assign it to the broadcast matrix.

The purpose of procedure \textit{HybridInitialization} as shown in Figure 13 is to initialize the
values of variables. First, in line 3, we compute the predictive number of slots per channel
on the average \((SC = \lceil \frac{P}{AC} \rceil)\). (Note that \(SC\) could be increased later.) In the previous
example as shown in Figure 2, we have \(SC = \lceil \frac{15}{3} \rceil = 5\). Next, in line 4, we order the
query sets from the hottest one to the coldest one based on their request ratios, and the
result is shown in Figure 2. Third, from 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 all pages have not 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 sum 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 access
frequency of each page is shown in Figure 5. Fourth, in lines 12 and 13, we order the pages
in each query set \((Q_i)\) from the hottest one to the coldest one and assign the result to the
 corresponding temporary query list, \(TQ_i\). For our previous example, the result of \(TQ_i\) is
shown in Figure 9. Finally, from 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 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 \textit{AssignPosition}(\(P_k, Q_r, ROW\)) is to find a non-conflicting slot
from row \(ROW\) of the broadcast matrix for a page \(P_k\) in query set \(Q_r\). In procedure
\textit{AssignPosition}, the value of \(NS_{ROW}\) represents the next available slot in channel \(ROW\),
and there may exist a non-conflicting slot for \(P_k\) from 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 will check the slots in \(AS_{ROW}\) first, if list \(AS_{ROW}\) is
not empty. That is because we prefer to assign a page to a slot near the beginning of the
01 Procedure HybridInitialization;
02 begin
03 $SC := \lfloor \frac{P}{Q} \rfloor$;
04 Sort $Q_i$ on their request ratios $R_i$ in a 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 a decreasing order and assign the result to $TQ_i$;
14 $X := \bigcup_i Q_i = \{P_1, P_2, ..., P_n\}$ be all broadcast data pages;
15 for $i := 1$ to $AC$ do
16 begin
17 $NS_i := 1$;
18 $AS_i := <>$;
19 end;
20 $ROW := 1$;
21 $COL := 1$;
22 end;

Figure 13: The HybridInitialization procedure

broadcast cycle. If there is no non-conflicting slot in $AS_{ROW}$ for page $P_k$, then we check the slots from $NS_{ROW}$. Moreover, if we still can not find a non-conflicting slot for page $P_k$ from slots $NS_{ROW}$ to $SC$, we will go down to check the slots in the next channel. If there is no non-conflicting slot in all available channels, we will assign page $P_k$ in 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 Figure 15 and Figure 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, it 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.
Procedure \textit{AssignPosition}(P_k; \text{integer}; Q_r; \text{set of pages}; \text{var} \ ROW; \text{integer});
begin
\( u := \text{TryAvailSlot}(ROW, Q_r); \) (* Step A *)
If \( u \neq -1 \) then (* Step B *)
begin (* there exists a non-conflicting slot before \( NS_{ROW} \) *)
\( M[ROW, u] := P_k; \)
\( F_k := 1; \)
\( AS_{ROW} := AS_{ROW} \{u\}; \)
end;
If \( F_k = 0 \) then (* all slots are conflicting before slot \( NS_{ROW} \) *)
begin
If \( NS_{ROW} > SC \) then (* Step C *)
begin (* Case 1: all slots of channel \( ROW \) have been checked *)
\( ROW := (ROW \mod AC) + 1; \) (* go down *)
\( C := C + 1; \)
end
else (* Case 2: not all slots of channel \( ROW \) have been checked *)
begin
\( COL := NS_{ROW}; \) (* Step D *)
while (CheckConflict(COL, Q_r)) and (COL \( \leq SC \)) (* Step E *)
begin (* the slot \( COL \) will induce a conflict *)
\( AS_{ROW} := AS_{ROW} + \{COL\}; \) (* record the available slot *)
\( COL := COL + 1; \) (* go right *)
end;
If \( COL > SC \) then (* Step F *)
begin (* out of the range of slots of the \( ROW \)th channel *)
\( NS_{ROW} := SC + 1; \)
\( ROW := (ROW \mod AC) + 1; \) (* go down *)
\( C := C + 1; \)
end
else (* Step G *)
begin (* a conflict does not occur *)
\( M[ROW, COL] := P_k; \)
\( F_k := 1; \)
\( NS_{ROW} := COL + 1; \)
end;
end; (* end of else *)
end; (* end of If \( F_k = 0 \) *)
If \( (F_k = 0) \) and \((C \geq AC)\) then (* Step H *)
begin (* all the channels have been checked *)
\( COL := NS_{AC}; \)
\( SC := SC + 1; \)
\( M[AC, COL] := P_k; \)
\( F_k := 1; \)
\( NS_{AC} := NS_{AC} + 1; \)
end;
end;

\textbf{Figure 14: The \textit{AssignPosition} procedure}
Function `TryAvailSlot(ROW: integer; Q_r: set of pages): integer;`
begin
  p := first(AS_{ROW}); (* p is the pointer to the list *)
  TryAvailSlot := -1;
  While (p ≠ nil) and (TryAvailSlot = -1) do
  begin
    Let TASE be the pth element of AS_{ROW};
    If (CheckConflict(TASE, Q_r) = False) then
    begin
      TryAvailSlot := TASE;
      AS_{ROW} := AS_{ROW} - < TASE >;
    end;
    p := next(AS_{ROW});
  end;
end;

Figure 15: The `TryAvailSlot` function

Function `CheckConflict(COL: integer; Q_r: set of pages): boolean;`
begin
  CheckConflict := False;
  i := 1;
  while (i ≤ AC) and (CheckConflict = False) do
  begin
    if (M[i, COL] ∈ Q_r) then CheckConflict := True;
    i := i + 1;
  end;
end;

Figure 16: The `CheckConflict` function

In procedure `AssignPosition(P_k, Q_r, ROW),` any input page P_k will go through step A to try to find a non-conflicting slot in list AS_{ROW} first. Next, an input page P_k will go through step B only if there is a non-conflicting slot in AS_{ROW}; otherwise, it will go through step C or step D. An input page P_k will go 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 will go 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 will 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 will process step F; otherwise, we will process step G. Finally, if there is no non-conflicting slot for page P_k in all available channels, we will process step H.
The purpose of function $\text{CheckConflict}(COL, Q_r)$ is to check whether the slot $COL$ is a conflicting slot for a page $P_k \ (\in Q_r)$. If the slot $COL$ in any available channel contains a page which is in the same query set $Q_r$ as page $P_k$, then the slot $COL$ is a conflicting slot for page $P_k \ (\in Q_r)$ and the resulting value of function $\text{CheckConflict}$ 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 $P_3$ in query set $Q_2$. We start to 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 $\text{CheckConflict}$ terminates with the value, $True$. For another example, when page $P_7$ in query set $Q_3$ is prepared 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 $\text{CheckConflict}$ terminates with the value, $False$.

The purpose of function $\text{TryAvailSlot}(ROW, Q_r)$ 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. (That is because we hope to assign a page at a more preceding slot as possible as we can.) Function $\text{TryAvailSlot}$ terminates when we find a non-conflicting slot. (Note that function $\text{TryAvailSlot}$ 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 elements in list $AS_{ROW}$, then function $\text{TryAvailSlot}$ terminates and returns -1. In function $\text{TryAvailSlot}$, we also call function $\text{CheckConflict}$ 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 that time, list $AS_2$ contains slot 1. Since slot 1 is not a conflicting slot for page $P_{11} \ (\in Q_4)$ (i.e., $\text{CheckConflict}(1, Q_4) = False$), function $\text{TryAvailSlot}$ 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 that time, list $AS_3$ contains slot 2. Since slot 2 will cause page $P_{12}$ to conflict with page $P_4$ (i.e., $\text{CheckConflict}(2, Q_5) = True$), function $\text{TryAvailSlot}$ returns -1.

Let’s use another example (Example 2) as shown in Figure 6 to show how the hybrid approach reduces the chance of waste of bandwidth. The tracing table of each page after processing procedure $\text{AssignPosition}$ is shown in Table 1 and the result of the broadcast
Data pages | $P_1$ | $P_2$ | $P_3$ | $P_5$ | $P_6$ | $P_{10}$ | $P_{12}$ | $P_8$ | $P_9$ | $P_{13}$ | $P_{15}$ | $P_{14}$
---|---|---|---|---|---|---|---|---|---|---|---|---
Access frequency | 92 | 68 | 55 | 55 | 35 | 32 | 25 | 25 | 20 | 8 | 8 | 7

Figure 17: The data pages of Example 2 after sorting on their frequencies

<table>
<thead>
<tr>
<th>Query list</th>
<th>Data pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TQ_1$</td>
<td>$P_1, P_2, P_3, P_4, P_5$</td>
</tr>
<tr>
<td>$TQ_2$</td>
<td>$P_1, P_2, P_6, P_{10}, P_{12}$</td>
</tr>
<tr>
<td>$TQ_3$</td>
<td>$P_1, P_3, P_4, P_8, P_9$</td>
</tr>
<tr>
<td>$TQ_4$</td>
<td>$P_2, P_7, P_{11}$</td>
</tr>
<tr>
<td>$TQ_5$</td>
<td>$P_1, P_8, P_{13}, P_{15}$</td>
</tr>
<tr>
<td>$TQ_6$</td>
<td>$P_1, P_8, P_{14}$</td>
</tr>
</tbody>
</table>

Figure 18: $TQ_i$ in Example 2 ($1 \leq i \leq NQ$)

matrix is shown in Figure 19. Note that after assigning page $P_{15}$ to $M[3,6]$, we have $SC = 6$ as shown in Table 1. That’s because there is no non-conflicting slot in all three channels for page $P_{15}$, page $P_{15}$ is assigned to the extended slot $(SC + 1) = 6$ of channel 3. Then we have $NS_3 = 7$ and update $SC = 6$. We extend the $3 \times 5$ broadcast matrix to a $3 \times 6$ matrix. Therefore, page $P_{14}$ can be considered to be assigned 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 processing steps of 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 $P_1$. For the next loop, after step $A$, we have $NS_1 (=6) \leq SC (=6)$; therefore, it will process step $D$. In step $D$, we have $COL = NS_1 = 6$. Since slot 6 is not a conflicting page for page $P_{14}$, it will skip step $E$. Moreover, we have $COL (=6) \leq SC (=6)$, so it will process step $G$. 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:

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

Figure 19: The output matrix of Example 2 based on the hybrid approach
Table 1: A tracing result of each page of Example 2 after procedure AssignPosition is processed

<table>
<thead>
<tr>
<th>Query List (r)</th>
<th>Page</th>
<th>ROW</th>
<th>COL</th>
<th>u</th>
<th>NS₁</th>
<th>NS₂</th>
<th>NS₃</th>
<th>AS₁</th>
<th>AS₂</th>
<th>AS₃</th>
<th>SC</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>TQ₁</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>&lt;&gt;</td>
<td>&lt;&gt;</td>
<td>&lt;&gt;</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>TQ₂</td>
<td>P₁</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>&lt;&gt;</td>
<td>&lt;&gt;</td>
<td>&lt;&gt;</td>
<td>5</td>
<td>M[1,1]</td>
</tr>
<tr>
<td>TQ₃</td>
<td>P₂</td>
<td>1</td>
<td>-1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>&lt;&gt;</td>
<td>&lt;&gt;</td>
<td>&lt;&gt;</td>
<td>5</td>
<td>M[1,2]</td>
</tr>
<tr>
<td>TQ₄</td>
<td>P₃</td>
<td>1</td>
<td>-1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>&lt;&gt;</td>
<td>&lt;&gt;</td>
<td>&lt;&gt;</td>
<td>5</td>
<td>M[1,3]</td>
</tr>
<tr>
<td>TQ₅</td>
<td>P₄</td>
<td>1</td>
<td>4</td>
<td>-1</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>&lt;&gt;</td>
<td>&lt;&gt;</td>
<td>&lt;&gt;</td>
<td>5</td>
<td>M[1,4]</td>
</tr>
<tr>
<td>TQ₆</td>
<td>P₅</td>
<td>1</td>
<td>5</td>
<td>-1</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>&lt;&gt;</td>
<td>&lt;&gt;</td>
<td>&lt;&gt;</td>
<td>5</td>
<td>M[1,5]</td>
</tr>
<tr>
<td>TQ₇</td>
<td>-</td>
<td>1</td>
<td>5</td>
<td>-1</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>&lt;&gt;</td>
<td>&lt;&gt;</td>
<td>&lt;&gt;</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>TQ₈</td>
<td>P₆</td>
<td>2</td>
<td>3</td>
<td>-1</td>
<td>6</td>
<td>4</td>
<td>1</td>
<td>&lt;&gt;</td>
<td>&lt;1&gt;</td>
<td>&lt;&gt;</td>
<td>5</td>
<td>M[2,3]</td>
</tr>
<tr>
<td>TQ₉</td>
<td>P₇</td>
<td>2</td>
<td>4</td>
<td>-1</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td>&lt;&gt;</td>
<td>&lt;1,2&gt;</td>
<td>&lt;&gt;</td>
<td>5</td>
<td>M[2,4]</td>
</tr>
<tr>
<td>TQ₁₀</td>
<td>P₈</td>
<td>2</td>
<td>5</td>
<td>-1</td>
<td>6</td>
<td>6</td>
<td>1</td>
<td>&lt;&gt;</td>
<td>&lt;1,2&gt;</td>
<td>&lt;&gt;</td>
<td>5</td>
<td>M[2,5]</td>
</tr>
<tr>
<td>TQ₁₁</td>
<td>-</td>
<td>3</td>
<td>5</td>
<td>-1</td>
<td>6</td>
<td>6</td>
<td>1</td>
<td>&lt;&gt;</td>
<td>&lt;1,2&gt;</td>
<td>&lt;&gt;</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>TQ₁₂</td>
<td>P₉</td>
<td>3</td>
<td>2</td>
<td>-1</td>
<td>6</td>
<td>6</td>
<td>3</td>
<td>&lt;&gt;</td>
<td>&lt;1,2&gt;</td>
<td>&lt;1&gt;</td>
<td>5</td>
<td>M[3,1]</td>
</tr>
<tr>
<td>TQ₁₃</td>
<td>P₁₀</td>
<td>3</td>
<td>5</td>
<td>-1</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>&lt;&gt;</td>
<td>&lt;1,2&gt;</td>
<td>&lt;1,3,4&gt;</td>
<td>5</td>
<td>M[3,5]</td>
</tr>
<tr>
<td>TQ₁₄</td>
<td>-</td>
<td>1</td>
<td>5</td>
<td>-1</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>&lt;&gt;</td>
<td>&lt;1,2&gt;</td>
<td>&lt;1,3,4&gt;</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>TQ₁₅</td>
<td>P₁₁</td>
<td>1</td>
<td>5</td>
<td>-1</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>&lt;&gt;</td>
<td>&lt;1,2&gt;</td>
<td>&lt;1,3,4&gt;</td>
<td>5</td>
<td>M[2,1]</td>
</tr>
<tr>
<td>TQ₁₆</td>
<td>P₁₂</td>
<td>3</td>
<td>5</td>
<td>-1</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>&lt;&gt;</td>
<td>&lt;1,2&gt;</td>
<td>&lt;1,4&gt;</td>
<td>5</td>
<td>M[3,3]</td>
</tr>
<tr>
<td>TQ₁₇</td>
<td>-</td>
<td>2</td>
<td>5</td>
<td>-1</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>&lt;&gt;</td>
<td>&lt;1,2&gt;</td>
<td>&lt;1,4&gt;</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>TQ₁₈</td>
<td>P₁₃</td>
<td>3</td>
<td>5</td>
<td>-1</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>&lt;&gt;</td>
<td>&lt;1,2&gt;</td>
<td>&lt;1&gt;</td>
<td>5</td>
<td>M[3,4]</td>
</tr>
<tr>
<td>TQ₁₉</td>
<td>P₁₄</td>
<td>3</td>
<td>6</td>
<td>-1</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>&lt;&gt;</td>
<td>&lt;1,2&gt;</td>
<td>&lt;1&gt;</td>
<td>6</td>
<td>M[3,6]</td>
</tr>
<tr>
<td>TQ₂₀</td>
<td>-</td>
<td>3</td>
<td>6</td>
<td>-1</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>&lt;&gt;</td>
<td>&lt;1,2&gt;</td>
<td>&lt;1&gt;</td>
<td>6</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 2: A case of page P₁₄ through steps ACADG

<table>
<thead>
<tr>
<th>loop</th>
<th>step</th>
<th>C</th>
<th>u</th>
<th>ROW</th>
<th>COL</th>
<th>NS₁</th>
<th>NS₂</th>
<th>NS₃</th>
<th>AS₁</th>
<th>AS₂</th>
<th>AS₃</th>
<th>SC</th>
<th>F₆</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>3</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>&lt;&gt;</td>
<td>&lt;&gt;</td>
<td>&lt;&gt;</td>
<td>&lt;2&gt;</td>
<td>&lt;1&gt;</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>0</td>
<td>-1</td>
<td>3</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>&lt;&gt;</td>
<td>&lt;2&gt;</td>
<td>&lt;&gt;</td>
<td>&lt;1&gt;</td>
<td>&lt;1&gt;</td>
</tr>
<tr>
<td>1</td>
<td>C</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>&lt;&gt;</td>
<td>&lt;2&gt;</td>
<td>&lt;&gt;</td>
<td>&lt;1&gt;</td>
<td>&lt;1&gt;</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>&lt;&gt;</td>
<td>&lt;2&gt;</td>
<td>&lt;&gt;</td>
<td>&lt;1&gt;</td>
<td>&lt;1&gt;</td>
</tr>
<tr>
<td>2</td>
<td>D</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>&lt;&gt;</td>
<td>&lt;2&gt;</td>
<td>&lt;&gt;</td>
<td>&lt;1&gt;</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>G</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>&lt;&gt;</td>
<td>&lt;2&gt;</td>
<td>&lt;&gt;</td>
<td>&lt;1&gt;</td>
<td>6</td>
</tr>
<tr>
<td>Query set</td>
<td>$Q_1$</td>
<td>$Q_2$</td>
<td>$Q_3$</td>
<td>$Q_4$</td>
<td>$Q_5$</td>
<td>$Q_6$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Access time</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>6</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Query set</th>
<th>$Q_1$</th>
<th>$Q_2$</th>
<th>$Q_3$</th>
<th>$Q_4$</th>
<th>$Q_5$</th>
<th>$Q_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access time</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

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

generated by the SNV strategy. Figure 20 and Figure 21 show the expected access time of each query set in the hybrid approach and the SNV strategy, respectively. The total expected access time based on the hybrid approach is as follows:

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

$$= (5 \times 35 + 5 \times 25 + 5 \times 20 + 3 \times 8 + 6 \times 7 + 6 \times 5)/100 = 4.96.$$  

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

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

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

Consequently, the total expected access time of the hybrid approach is less than that of 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 in
the generation of the synthetic query sets are shown in Table 3 [4]. The length of a query set is determined by a Poisson distribution with mean \( \mu \) 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 at least in a 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 previous 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 \( C \), there are \( |P| \times F\% \) 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 previous generated query set.

2. If all pages have occurred in previous generated query sets \( P \), the remaining data pages are picked at random form \( D \). Then, we will check whether the query set \( C \), which is being generated, is the same as any previous 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 previous generated query \( P \).

The request ratio of each query set is picked from an exponential distribution with mean equal to 1. The request ratios are normalized such that the sum of all 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 those 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 those 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

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(D)</td>
<td>Number of data pages in database</td>
</tr>
<tr>
<td>(</td>
<td>\text{MeanQ}</td>
</tr>
<tr>
<td>(</td>
<td>\text{MinQ}</td>
</tr>
<tr>
<td>(</td>
<td>\text{MaxQ}</td>
</tr>
<tr>
<td>(NQ)</td>
<td>Number of query sets</td>
</tr>
<tr>
<td>correlation level</td>
<td>The parameter to decide the similar fraction for pages in the adjacent query set</td>
</tr>
</tbody>
</table>

4.2 Performance Analysis

For the total number of slots (denoted as \(TS\)) in a broadcast cycle in the SNV strategy, it can be computed as follows:

\[
TS = NS - 1.
\]

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

\[
TS = NS_3 - 1 = 8 - 1 = 7.
\]

While the total number of slots in a broadcast cycle in the hybrid approach, it can be computed as follows:

\[
TS = SC = NS - 1.
\]

Take Example 2 as shown in Figure 6 for example, the result of broadcast matrix based on the SNV strategy is shown in Figure 19 and its \(TS\) can be calculated as follows:

\[
TS = SC = 6 = NS_3 - 1 = 7 - 1.
\]

Moreover, the ways to calculate the total number of empty slots (denoted as \(TES\)) in a broadcast cycle in the SNV strategy and in 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 (as shown in Figure 6) for 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 ways to calculate the total expected delay time (denoted as \(TEAT\)) in the SNV strategy and in the hybrid approach are the same.

The total expected delay time can be computed as follows:
\[ T_{EAT} = \sum_{\tau=1}^{N_Q} R_{\tau} \times AT_{\tau}. \]

There are two policies to calculate \( AT_{\tau} \).

Let \( SPN_{i}^r \) be the number of pages \((\in Q_\tau)\) which are assigned to slot \( i \) and let \( MaxSPN \) be the maximal value of \( SPN_{i}^r \), \( 1 \leq i \leq (NS_{AC} - 1) \). If \( MaxSPN = 1 \), it indicates 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, it indicates that the whole query set cannot be retrieved in a broadcast cycle. Therefore, we can calculate \( AT_{\tau} \) by the following policies:

1. If the maximal value of \( SPN_{i}^r \) (called \( MaxSPN \)) equals to 1, \( 1 \leq i \leq (NS_{AC} - 1) \), then \( AT_{\tau} \) is the maximal value of \( i \) with \( SPN_{i}^r = 1 \).

2. If \( MaxSPN > 1 \), \( 1 \leq i \leq (NS_{AC} - 1) \), we let \( MP \) be the maximal value of \( i \) which has \( SPN_{i}^r = MaxSPN \). Then, \( AT_{\tau} \) can be computed as follows:
\[
AT_{\tau} = TS \times (MaxSPN - 1) + MP.
\]

Take Example 2 shown in Figure 6 for example, based on the \( SNV \) strategy, the result of the broadcast matrix is shown in Figure 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 \( SPN_1^1 = SPN_2^2 = SPN_7^7 = 1 \) and \( SPN_6^3 = SPN_6^4 = SPN_6^5 = SPN_6^6 = 0 \). The maximal value of \( SPN_{i}^r \) is 1. Therefore, the total access time of query set \( Q_6 \) in the \( SNV \) strategy which is denoted as \( AT_6 \) equals to 7 that 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 Figure 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 \( SPN_1^1 = SPN_2^2 = SPN_6^6 = 1 \) and \( SPN_6^3 = SPN_6^4 = SPN_6^5 = 0 \). The maximal value of \( SPN_{i}^r \) is 1. Therefore, the total access time of query set \( Q_6 \) in the \( SNV \) strategy which is denoted as \( AT_6 \) equals to 6 that 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 make a comparison with Ke's \( SNV \) strategy [22]. In this simulation, we let \( D = 800 \), \( correlation = 0.5 \) and \( |MinQ| = 1 \). We consider 27 test samples which include the combinations of \( |MeanQ| = \)
Table 4: The parameters and their default settings used in the simulation of multiple channel

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D$</td>
<td>800</td>
</tr>
<tr>
<td>$</td>
<td>\text{MeanQ}</td>
</tr>
<tr>
<td>$</td>
<td>\text{MinQ}</td>
</tr>
<tr>
<td>$</td>
<td>\text{MaxQ}</td>
</tr>
<tr>
<td>$NQ$</td>
<td>200, 300, 400</td>
</tr>
<tr>
<td>$\text{correlation level}$</td>
<td>0.5</td>
</tr>
<tr>
<td>$AC$</td>
<td>3, 5, 7</td>
</tr>
</tbody>
</table>

7, 9 and 11, $|\text{MaxQ}| = 2 \times |\text{MeanQ}|$, $NQ = 200, 300$ and $400$, and $AC = 3, 5$ and $7$. For each test sample, we compute the average result for executing 100 times. The parameters and their default settings are shown in Table 4. To ensure that all 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: $NQ \times |\text{MeanQ}| \geq D$.

A broadcast cycle starts from the first slot and ends to 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 $N_{SAC} - 1$ in the SNV strategy and in the hybrid approach. Since the SNV strategy only extend the number of slots in the last channel, $NS_{AC}$ in the SNV strategy is greater than that in the hybrid approach. Obviously, the total number of slots in the SNV strategy is greater than that in the hybrid approach. When $NQ = 200, 300$ and $400$, the detailed simulation results about the total number of slots in a broadcast cycle in the hybrid approach and the SNV strategy for 100 executions are shown in Tables 5, 6 and 7, respectively, where Hybrid denotes our proposed hybrid approach, SNV denotes Ke’s SNV strategy. From the results, we show that our hybrid approach always generates a smaller number of slots than the SNV strategy. As $AC$ is increased, the total number of slots is decreased in both the hybrid approach and the SNV strategy. As $|\text{MeanQ}|$ is increased, the total number of slots is increased in both the hybrid approach and the SNV strategy.

The total number of empty slots in a broadcast cycle equals to $(TS \times AC - D)$. Since $TS$ in the SNV strategy is usually greater than that in the hybrid approach, obviously, the
Table 5: A comparison of total number of slots in a broadcast cycle ($D = 800$, $NQ = 200$)

<table>
<thead>
<tr>
<th>MeanQ</th>
<th>Hybrid</th>
<th>SNV</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$AC = 3$</td>
<td>$AC = 5$</td>
</tr>
<tr>
<td>7</td>
<td>268.51</td>
<td>163.77</td>
</tr>
<tr>
<td>9</td>
<td>269.06</td>
<td>164.66</td>
</tr>
<tr>
<td>11</td>
<td>269.76</td>
<td>165.18</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>MeanQ</th>
<th>Hybrid</th>
<th>SNV</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$AC = 3$</td>
<td>$AC = 5$</td>
</tr>
<tr>
<td>7</td>
<td>268.57</td>
<td>163.85</td>
</tr>
<tr>
<td>9</td>
<td>269.02</td>
<td>164.46</td>
</tr>
<tr>
<td>11</td>
<td>269.72</td>
<td>165.11</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>MeanQ</th>
<th>Hybrid</th>
<th>SNV</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$AC = 3$</td>
<td>$AC = 5$</td>
</tr>
<tr>
<td>7</td>
<td>268.60</td>
<td>163.71</td>
</tr>
<tr>
<td>9</td>
<td>269.08</td>
<td>164.49</td>
</tr>
<tr>
<td>11</td>
<td>269.62</td>
<td>165.45</td>
</tr>
</tbody>
</table>
Table 8: A comparison of total number of empty slots in a broadcast cycle \((D = 800, NQ = 200)\)

<table>
<thead>
<tr>
<th>MeanQ</th>
<th>Hybrid</th>
<th>SNV</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(AC = 3)</td>
<td>(AC = 5)</td>
</tr>
<tr>
<td>7</td>
<td>5.53</td>
<td>18.85</td>
</tr>
<tr>
<td>9</td>
<td>7.18</td>
<td>23.30</td>
</tr>
<tr>
<td>11</td>
<td>9.28</td>
<td>25.90</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>MeanQ</th>
<th>Hybrid</th>
<th>SNV</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(AC = 3)</td>
<td>(AC = 5)</td>
</tr>
<tr>
<td>7</td>
<td>5.71</td>
<td>19.25</td>
</tr>
<tr>
<td>9</td>
<td>7.06</td>
<td>22.30</td>
</tr>
<tr>
<td>11</td>
<td>9.16</td>
<td>25.55</td>
</tr>
</tbody>
</table>

total number of empty slots in a broadcast cycle in the hybrid approach is greater than that in the \(SNV\) strategy. When \(NQ = 200, 300\) and \(400\), the detailed simulation results about the total number of empty slots in a broadcast cycle in the hybrid approach and the \(SNV\) strategy for 100 executions are shown in Tables 8, 9 and 10, respectively. From the results, we show that our hybrid approach always generates a smaller number of empty slots than the \(SNV\) strategy. As \(AC\) is increased, the total number of empty slots is increased in both the hybrid approach and the \(SNV\) strategy. As \(|MeanQ|\) is increased, the total number of empty slots is increased in both the hybrid approach and the \(SNV\) strategy.

Table 10: A comparison of total number of empty slots in a broadcast cycle \((D = 800, NQ = 400)\)

<table>
<thead>
<tr>
<th>MeanQ</th>
<th>Hybrid</th>
<th>SNV</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(AC = 3)</td>
<td>(AC = 5)</td>
</tr>
<tr>
<td>7</td>
<td>5.80</td>
<td>18.55</td>
</tr>
<tr>
<td>9</td>
<td>7.24</td>
<td>22.45</td>
</tr>
<tr>
<td>11</td>
<td>8.86</td>
<td>27.25</td>
</tr>
</tbody>
</table>
Table 11: A comparison of total expected access time \( (D = 800, NQ = 200) \)

<table>
<thead>
<tr>
<th>MeanQ</th>
<th>Hybrid ( AC = 3 )</th>
<th>Hybrid ( AC = 5 )</th>
<th>Hybrid ( AC = 7 )</th>
<th>SNV ( AC = 3 )</th>
<th>SNV ( AC = 5 )</th>
<th>SNV ( AC = 7 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>43.167</td>
<td>27.822</td>
<td>21.320</td>
<td>43.990</td>
<td>28.745</td>
<td>22.310</td>
</tr>
<tr>
<td>9</td>
<td>53.406</td>
<td>34.309</td>
<td>26.831</td>
<td>54.413</td>
<td>35.407</td>
<td>28.023</td>
</tr>
<tr>
<td>11</td>
<td>64.169</td>
<td>41.493</td>
<td>32.065</td>
<td>65.382</td>
<td>42.876</td>
<td>33.414</td>
</tr>
</tbody>
</table>

Table 12: A comparison of total expected access time \( (D = 800, NQ = 300) \)

<table>
<thead>
<tr>
<th>MeanQ</th>
<th>Hybrid ( AC = 3 )</th>
<th>Hybrid ( AC = 5 )</th>
<th>Hybrid ( AC = 7 )</th>
<th>SNV ( AC = 3 )</th>
<th>SNV ( AC = 5 )</th>
<th>SNV ( AC = 7 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>43.245</td>
<td>27.822</td>
<td>21.278</td>
<td>44.134</td>
<td>28.782</td>
<td>22.341</td>
</tr>
<tr>
<td>9</td>
<td>53.377</td>
<td>34.535</td>
<td>26.751</td>
<td>54.419</td>
<td>35.630</td>
<td>27.940</td>
</tr>
<tr>
<td>11</td>
<td>64.702</td>
<td>41.695</td>
<td>31.862</td>
<td>65.848</td>
<td>42.897</td>
<td>33.246</td>
</tr>
</tbody>
</table>

When \( NQ = 200, 300 \) and \( 400 \), the detailed simulation results about the total expected access time in the hybrid approach and the SNV strategy for 100 executions are shown in Tables 11, 12 and 13, respectively. From the results, we show that our hybrid approach always requires a shorter expected access time than the SNV strategy. As \( AC \) is increased, the total expected access time is decreased in both the hybrid approach and the SNV strategy. As \( \text{MeanQ} \) is increased, the total expected access time is increased in both the hybrid approach and the SNV strategy.

When \( NQ = 200, 300 \) and \( 400 \), the detailed simulation results about the average expected access time of each query set in the hybrid approach and the SNV strategy for 100 executions are shown in Tables 14, 15 and 16, respectively. From the results, we show that

Table 13: A comparison of total expected access time \( (D = 800, NQ = 400) \)

<table>
<thead>
<tr>
<th>MeanQ</th>
<th>Hybrid ( AC = 3 )</th>
<th>Hybrid ( AC = 5 )</th>
<th>Hybrid ( AC = 7 )</th>
<th>SNV ( AC = 3 )</th>
<th>SNV ( AC = 5 )</th>
<th>SNV ( AC = 7 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>42.762</td>
<td>27.841</td>
<td>21.384</td>
<td>43.638</td>
<td>28.794</td>
<td>22.381</td>
</tr>
<tr>
<td>9</td>
<td>53.280</td>
<td>34.678</td>
<td>26.530</td>
<td>54.279</td>
<td>35.828</td>
<td>27.605</td>
</tr>
<tr>
<td>11</td>
<td>64.591</td>
<td>41.573</td>
<td>31.847</td>
<td>65.830</td>
<td>42.849</td>
<td>33.217</td>
</tr>
</tbody>
</table>
Table 14: A comparison of average expected access time of each query set \((D = 800, NQ = 200)\)

<table>
<thead>
<tr>
<th>MeanQ</th>
<th>Hybrid</th>
<th>SNV</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AC = 3</td>
<td>AC = 7</td>
</tr>
<tr>
<td>7</td>
<td>169.39</td>
<td>76.79</td>
</tr>
<tr>
<td>9</td>
<td>189.76</td>
<td>89.14</td>
</tr>
<tr>
<td>11</td>
<td>207.11</td>
<td>100.00</td>
</tr>
</tbody>
</table>

Table 15: A comparison of total expected access time \((D = 800, NQ = 300)\)

<table>
<thead>
<tr>
<th>MeanQ</th>
<th>Hybrid</th>
<th>SNV</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AC = 3</td>
<td>AC = 7</td>
</tr>
<tr>
<td>7</td>
<td>192.93</td>
<td>88.45</td>
</tr>
<tr>
<td>9</td>
<td>210.70</td>
<td>100.13</td>
</tr>
<tr>
<td>11</td>
<td>227.36</td>
<td>110.88</td>
</tr>
</tbody>
</table>

Our hybrid approach always requires a shorter expected access time than the SNV strategy. As \(AC\) is increased, the average expected access time of each query set is decreased in both the hybrid approach and the SNV strategy. As |MeanQ| is increased, the average expected access time of each query set is increased in both the hybrid approach and the SNV strategy.

Table 16: A comparison of average expected access time of each query set \((D = 800, NQ = 400)\)

<table>
<thead>
<tr>
<th>MeanQ</th>
<th>Hybrid</th>
<th>SNV</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AC = 3</td>
<td>AC = 7</td>
</tr>
<tr>
<td>7</td>
<td>204.30</td>
<td>94.35</td>
</tr>
<tr>
<td>9</td>
<td>221.98</td>
<td>105.87</td>
</tr>
<tr>
<td>11</td>
<td>237.82</td>
<td>116.55</td>
</tr>
</tbody>
</table>
5 Conclusion

In many situations, a mobile client might need data of more than one page. Moreover, as a number of wireless channels are available, the design of scheduling strategies becomes more difficult. In this paper, we have proposed an efficient hybrid approach to query sets broadcast scheduling for multiple channels. From our performance analysis and simulation, we have shown that our hybrid approach requires shorter total expected delay access time, and creates smaller number of total slots and smaller number of empty slots in one broadcast cycle than the SNV strategy. In an asymmetric environment, when a client receives an information item containing errors (due to some environment 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 important concern. Therefore, how to efficient design broadcast programs in the real-time environment is the possible future research direction.

References


