國立中興大學資訊科學與工程學系

碩士學位論文

在無線環境中以查詢集為基礎之廣播資料排程

Query-Set-Based Scheduling of Data Broadcast in the Wireless Environments

指導教授：賈坤芳

研 究 生：林大中

中華民國九十七年六月

國立中興大學 資訊科學與工程 研究所

碩士學位論文

題目：在無線環境中以查詢集為基礎之廣播資料排程

姓名：林 大 中 學號：79556018

經 口 試 通 過 特 此 證 明

論文指導教授 賈坤芳

論文考試委員

[illegible]

楊東麟

[illegible]

賈坤芳

中華民國 97 年 6 月 24 日

ACKNOWLEDGEMENTS

Thanks be to God for His indescribable gift in the past two year. I would like to thank all people for accompanying me, and your support means a lot. The deeply gratitude goes first to Dr. Kuen-Fang Jea, my advisor, for his constant guidance and encouragement. Without his leading, I cannot find out the entrance to the science world. Special thanks to Jen-Ya Wang for his suggestion and assistance in my study. Furthermore, let me say 'thank you' to the member of Database and System software Lab. It is my pleasure to meet you all. At last, I would thank my family: my parents, brother, and sister, for unconditional support and unceasing intercession before God.

Because of the merciful compassions of our God, in which the rising sun will visit us from on high, to shine upon those sitting in darkness and in the shadow of death, to guide our feet into the way of peace.

摘要

近年來無線技術的應用快速增加，越來越多的行動用戶在日常生活中仰賴其行動運算裝置，來取得所需的資訊，例如行動電話或個人數位助理。廣播是一種有效且可擴展的資料傳輸方法，能將同一資料傳輸給無限數量的行動用戶。為了要應付這種龐大的資料傳輸量以滿足所有行動用戶的需求，資料廣播的應用一直是許多人注意的熱門技術。其中一項重要議題是如何在無線環境的廣播頻寬限制下，能夠快速回答使用者的查詢。因此，目前有許多研究著手於如何在單一頻道或多頻道的環境下，產生最佳廣播資料排程，降低所有行動用戶的平均存取時間或平均預計延遲時間。在本研究中，我們利用現有最佳資料項複製排程的公式，推導出最佳查詢集複製排程的方程式，並且利用查詢集複製的概念，提出一種以查詢集為基礎的廣播資料排程演算法，稱作 QSBS。QSBS 排程演算法將同一查詢集中的所有資料項儘量放置於同一個頻道上，並且儘可能的將其聚集在一起播送，其結果因此降低了所有行動用戶的平均存取時間。最後，本研究實作了 QSBS 以及其它作為比較對象的演算法，並實際進行模擬實驗，評估各排程演算的法效能。

關鍵詞：行動運算、查詢集為基礎、資料排程、資料廣播、資料複製

Abstract

In the recent years, the applications of wireless technology have rapidly grown. There are more and more mobile users who obtain information with mobile devices (e.g., cell phone, or PDA) in their daily life. Broadcast is an efficient and scalable way of transmitting data to an unlimited number of mobile users. In order to cope with a huge amount of data access for satisfying all mobile users' requests, data broadcast becomes popular and attracts researchers' attention. One important issue in data broadcast is how to quickly service users' request queries under the limited bandwidth of the wireless environment; therefore, researchers have focused on generating an optimal broadcast program to minimize the average access or expected delay time for all mobile users in a single or multiple channels environments. In this study, an equation of the optimal broadcast program with query-set replication is derived from the optimal broadcast program with data-item replication in a previous research. Then, a query-set-based scheduling algorithm, named QSBS, for producing broadcast program is proposed based on the concept of query-set replication. QSBS will make all data items in a query set be allocated in the same channel and be broadcast as near as possible. In this way, the average access time for all mobile users can be decreased. Finally, we implement QSBS and other broadcast-program scheduling algorithms, and conduct experiments to compare their performance.

Keywords: Mobile computing, Query-set-based access, Data scheduling, Data broadcast, Data-item replication

Table of Contents

Chapter **Page**

List of Figures

Figure **Page**

List of Tables

Table **Page**

1 Introduction

As the technologies of mobile devices and wireless networks have been developed rapidly, many applications have become possible; for example, people can enjoy the video communication, know the real-time traffic information or connect to the internet by their mobile devices anytime anywhere. However, there exist several restrictions on mobile computing. For instance, the limited battery capacity requires both lower computing power and less standby time of mobile devices, and the bandwidth limitation of physical communications medium makes slower speed for data transmission.

Broadcast is an efficient and scalable way of transmitting data to an unlimited number of mobile users. Through disseminating all data items in broadcast channels, every mobile user can tune into the channels and retrieve his/her interested data arbitrarily. Therefore, many researchers have paid attention to this technology. In order to utilize the limited resources of power and bandwidth, [1] proposes a data dissemination architecture in which a server continuously and repeatedly broadcasts the program of scheduled data items to a group of mobile users through a single broadcast channel. There are three categories of data broadcast: (1) pure-push-based [1, 14, 24, 33, 34], (2) pure-pull-based [3-5], and (3) hybrid-based [2, 8, 22, 31]. The pure-push-based technique is that the server allocates all data items in the database by a specified algorithm to broadcast channels and transmits data items periodically. All mobile users listen to the broadcast channels passively and retrieve their interested data items. The pure-pull-based technique requires that mobile users have to send requests to the server. Each request is processed by the server via an on-demand channel which is one of the available broadcast channels. The hybrid-based technique is a compound approach of pure-push-based technique and pure-pull-based technique; moreover, it intends to

benefit both the server and the mobile users.

In the early time, most of researches concerning data broadcast assume that each mobile user requests only one data item every time, and all data items are independent of each other. Recently, however, more complex applications make the dependency among data items, and mobile users may issue a request of multiple data items. For example, a mobile user would like to search all articles with keyword, *broadcast*, in the libraries located in Taipei, Tokyo and New York, respectively. If a server treats this request as three independent requests, the mobile user has to send three requests, and the requests are processed individually. Thus, an article which exists in these three libraries simultaneously will be transmitted three times, and the mobile user will spend more time on retrieving the redundant results.

Data scheduling is the problem of allocating data items to a single or multiple channels. According to the context of all user-issued query sets, a broadcast server will generate a broadcast program to minimize the average access time with a specific scheduling algorithm. In this study, a query-set-based scheduling algorithm named QSBS for producing broadcast programs is proposed. Each mobile user issues requests of multiple data items, and a query set is made up of all data items that a mobile user requests. According to all of the query sets, QSBS produces a broadcast program with data-item replication over multiple broadcast channels. There are mainly three features in our study. First, a query set is treated as a *compound data item*. With this novel viewpoint of a query set, a near-optimal equation with query-set replication and the conditions of replication are derived. Second, the novel concept of query set employed in algorithm QSBS replicates data items among broadcast channels. However, the proper replicas of data items are good for better degree of coherence for a query set. Third, algorithm QSBS makes all data items in a query set be allocated in the same

channel as close as possible; therefore, a mobile user needs not to switch channels frequently and can retrieve all data items from the least number of channels.

The rest of this thesis is organized as follows. Chapter 2 reviews the related work. The definitions of the data scheduling problem, assumptions and examples are proposed in Chapter 3. In Chapter 4, the query-set-based concept and the algorithm are illustrated. Experimental results and analyses are showed in Chapter 5. Chapter 6 is the conclusion and future work of this research.

2 Related work

Among the techniques of mobile computing, data broadcast has attracted much attention of many researchers. Usually, the average access time and the average expected delay time are two performance metrics for the broadcast-program scheduling algorithm. The access time is defined as the total time that a mobile user has to spend on retrieving all the desired one or more data items, and the average access time is computed as the sum of all access times, weighted by their access probabilities. However, mobile users may care about the time that they have to spend on receiving unwanted data items before retrieving the desired ones. The time that mobile users waste in tuning into a broadcast channel to start retrieving desired data items is called expected delay time. The average expected delay time is computed as the sum of all expected delay times, weighted by their access probabilities.

In the beginning, the server simply constructs a flat broadcast program and broadcasts all data items cyclically, while each data item will be transmitted one time during a broadcast cycle. In this approach, the access probabilities of all data items are assumed to be identical, and the access time for retrieving each data item is identical. However, the access time will increase when the distribution of access probability is skewed. Acharya *et al.* [1] propose an approach named *broadcast disk*, which takes into consideration the skewed access probabilities of all data items. In the broadcast disk approach, a single broadcast channel is used to broadcast data items in different frequencies according to their access probabilities. That is, more popular data items should be broadcasted more frequently.

With data-item replication, Wong [37] derives the theoretical lower bound of average access time for the requests of a single data item of the same size in a single

channel. The optimal conditions, however, are hard to implement in real applications where data items are of variable size and multiple broadcast channels are employed. Hameed and Vaidya [14, 15, 35] extend the lower bound for data items of variable size and propose an approximate scheduling algorithm with bucketing in multiple broadcast channels. Besides, Kang *et al.* [28] propose the multi-frequency scheduling (MFS) based on the concept of data-item replication. They partition all data items into three sets, hot, cold and non-outstanding, according to the access probability of each data item. With a limited number of replicas and approximately spaced, a data item in the hot set will have more replicas in the broadcast program according to its access probability.

Since deciding the optimal number of replicas of a data item is a difficult problem, several approaches without data-item replication are proposed. [32] transforms data scheduling into a channel allocation tree with variant-fanout, and proposes a heuristic algorithm VF^K. [30] proposes a heuristic with directed acyclic graph (DAG). The two heuristics above are based on dynamic programming which needs to keep track of prior and partial solutions to yield the optimal solution. Therefore, they are time consuming to construct the broadcast program. [7, 16, 26, 36, 39] transform the scheduling problem into a partition problem. They partition all data items into several groups and broadcast each group in each channel according to the access probability of each data item. In [39], an optimal partition condition is proposed, and the data partition with dynamic programming and greedy method is implemented. Based on the optimal partition, [7] proposes two pseudo-polynomial algorithms, one for data item of equal size and the other for that of variable size, [26] proposes a polynomial time algorithm with divide-and-conquer method, and [36] proposes a restricted dynamic programming algorithm. The average expected times of algorithms in [7, 26, 36, 39] are better than those of the algorithms with dynamic programming. Hsu *et al.* [16] use the optimal

scheduling with data-item replication proposed in [37] to compute the length of each partition group, and propose a near-optimal heuristic algorithm. Its access time is close to the lower bound of average access time.

The approaches above focus on the requests of a single data item; however, mobile users may issue the requests of multiple data items in complex applications. That is, a mobile user requests more than one data item in a query set at a time. A query set is made up of the user-requested data items. The simplest query set is the query set of two data items. An optimal scheduling in a single broadcast channel for requesting two data items of equal and variable size is proposed in [9].

Concerning more data items in query sets, algorithm GA is proposed in [18-21], which addresses the problem of broadcasting query sets of multiple data items in multiple channels with data-item replication. Algorithm GA employs the Genetic Algorithm [13]. Basically, algorithm GA is composed of iterative procedures that search the problem solutions by an evolutionary process based on natural selection; however, the iterative procedures of algorithm GA cost much processing time to generate the broadcast program.

Chung *et al.* [12] propose a broadcast scheduling method without data-item replication called *Query Expansion Method* (QEM) to minimize the average access time. A measure, named *query distance* (QD), is used to represent the degree of data-item's coherence for a query set. The access time for a query set can be minimized as long as the QD of a query set can be minimized, i.e., all data items in a query set are located coherently in a broadcast program. A query set with higher access probability deserves a higher priority to be scheduled and has the better QD of it. Algorithm QEM attempts to minimize the total QD among all query sets. Based on QEM, Lee and Lo [29] present an algorithm, named MQEM, with some modifications to improve algorithm QEM. In

algorithm QEM, the result of scheduled query sets cannot be changed, but algorithm MQEM neglects this restriction. Algorithm MQEM considers that the total QD may be minimized when the next query set is scheduled into the previous result of broadcast program. Moreover, some researches focus on minimizing QD by association-rule mining [11] or by the eigenvector optimization method [38].

Algorithm PBA [25], standing for Placement-Based Allocation, is a placement-based algorithm without data-item replication in multiple channels. The query sets with higher access probabilities have higher priorities to be scheduled by PBA; however, a query set of lower priority has a lower degree of coherence in the broadcast program. Based on algorithm PBA, the framework named MULS is proposed in [23]. The query sets in algorithm PBA are unordered, and those in MULS are ordered. A mobile user who produces an ordered query set should retrieve all data items in a fixed order, whereas a mobile user who issues an unordered query set is allowed to retrieve all data items in an arbitrary order.

According to the current research, there exists an optimal access time for a query set when all data items in the query set are scheduled coherently in a broadcast program. It motivates this research to treat a query set as a compound data item, and to schedule all query sets (compound data items) into multiple broadcast channels. By the viewpoint of the compound data item, the optimal conditions with query-set (compound-data-item) replication can be derived. In this thesis, this concept is employed in designing a novel query-set-based algorithm for generating a broadcast program.

3 Preliminaries

3.1. Problem definitions and assumptions

The problem to be solved in this thesis is *reducing the average access time for mobile users who request for retrieving multiple data items in query sets in the multi-channel environment*.

According to the prior researches [12, 25, 37], there are three significant factors affecting the average access time for all mobile users. First, mobile users will spend less access time on retrieving data items in a shorter broadcast cycle. Second, all data items in a query set scheduled coherently in a single broadcast channel will cause less access time for the query set. Third, more replicas of a data item are good for retrieving the data item in a single or multiple channels. However, there are also three negative factors. First, a broadcast cycle is decided by the number of data items in a broadcast database, i.e., the more data itmes cause the longer broadcast program. Second, it is hard to schedule all data items coherently for each query set every time. Third, the access time for retrieving data items will increase as one of them has more replicas. Hence, a query-set-based gerenting algorithm is proposed in this thesis. All data items in a query set will be scheduled as coherently as possible, and mobile users can retireve all desired data items from the least number of broadcast channels.

The symbol and assumptions in this research are described as follows. Assume that a broadcast server is equipped with K broadcast channels, and there are n data items of equal size in the broadcast database $D = \{d_1, d_2, \ldots, d_n\}$, $n \geq 1$. Each mobile user issues a query set containing multiple data items, and the server collects all query sets to serve all mobile users' requests. A query set q_j requested by any mobile user is made up of n_j ($2 \leq n_j < n$) data items in D, and it is assumed *unordered*. That is, all data items in a

query set can be retrieved in arbitrary order by mobile users. A query profile Q is defined as a collection of all query sets with their corresponding access probabilities, i.e., $Q = \{(q_1, P_1), (q_2, P_2), \ldots, (q_\lambda, P_\lambda)\}$, where λ ($\lambda \geq 1$) is the number of the pairs, and (q_j, P_j) represents that the access probability of q_j is P_j. Besides, each q_j and P_j in Q has to satisfy $\bigcup_{j=1}^{\lambda} q_j = D$ and $\sum_{j=1}^{\lambda} P_j = 1$. According to the context of Q, the broadcast server generates a broadcast program Φ with data items replicated over K broadcast channels by using a scheduling algorithm. A data item scheduled into a broadcast channel i may be replicated in other channels, but not in channel i. The server delivers data items periodically, and all mobile users retrieve desired data items from Φ. Suppose that the slotted time model [10, 17] is employed in this thesis, where the broadcast server takes one time slot to broadcast each data item in D. Therefore, the two terms, time and data slots, are used interchangeably in this thesis. In the slotted time model, a mobile user is assumed to tune into a broadcast channel at the beginning of a data slot and to leave at the end of a data slot, i.e., there are $|\Phi|$ possible moments at which mobile users may tune into the broadcast program Φ. The average access time for retrieving all data items in a query set q_j, denoted by $T_{Avg}(q_j)$, is defined as the average time which a mobile user spends on receiving all the data items in q_j from Φ. In the slotted time model, $T_{Avg}(q_j)$ equals the sum of the access time for q_j at all possible moments divided by $|\Phi|$. Table 3.1 lists the parameters described above and their descriptions.

Table 3.1. Description of the parameters

Parameters	Descriptions
d_i	The i-th data item.

D	The set of data item d_i, $D = \{d_1, d_2, \ldots, d_n\}$.
n	The total number of data items in D.
q_j	The j-th query set, $q_j = \left\{d_1^j, d_2^j, \ldots, d_{n_j}^j\right\}$.
d_x^j	The x-th data item in q_j, $1 \le j \le \lambda$, $1 \le x \le n_j$.
n_j	The number of data items in query set q_j.
Q	The set of q_j, $Q = \{(q_1, P_1), (q_2, P_2), \ldots, (q_\lambda, P_\lambda)\}$.
λ	The number of query sets in Q.
P_j	The access probability of the query set q_j.
p_i	The access probability of the data item d_i.
K	The total number of broadcast channels.
Φ	A broadcast program.
$\|\Phi\|$	The cycle length of Φ.

Definition 1. The average access time for multiple query sets in a query profile

Given a non-empty database D, a query profile Q and a scheduled broadcast program Φ, the average access time $T_{Avg}(Q)$ for all query sets q_j in Q from Φ is defined as

$$T_{Avg}(Q) = \sum_{j=1}^{\lambda} T_{Avg}(q_j) \times P_j \tag{3.1}$$

Example 1. The average access time for a single query set

Given a database $D = \{d_1, d_2, d_3, d_4, d_5\}$, the five data items in D is assumed to have been scheduled into a broadcast channel. Suppose that a specific broadcast program for D is, $\Phi = [d_1, d_2, d_3, d_4, d_5]$, as shown in Figure 3.1 (The five arrows A to E

indicate the starting moments of the five data slot, respectively). Because of the cyclic broadcasting, the five data items will be transmitted in the sequential order, $d_1 \rightarrow d_2 \rightarrow d_3 \rightarrow d_4 \rightarrow d_5 \rightarrow d_1 \rightarrow d_2 \rightarrow$ …, etc.

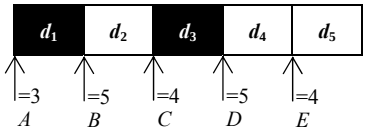


Figure 3.1. An example of computing average access time.

Assume that a query set, $q = \{d_1, d_3\}$, is issued by a mobile user. For the sake of illustration, we assume that a mobile user tunes into the broadcast channel at the moment A in Figure 3.1. Therefore, the mobile user will spend three time slots on retrieving d_1 and d_3. The mobile user, although not needing the data item d_2, still has to spend one time slot on waiting for d_2 to pass over. Furthermore, suppose that the mobile user tunes in at the moment B, he/she will spend five time slots, two on receiving d_1 and d_3, and three on waiting for d_2, d_4 and d_5 to pass over. By this way, the mobile user has to spend four time slots at C, i.e., five time slots at D, and four time slots at E, respectively. Therefore, the average access time for this mobile user to receive all the data items of q from Φ is, $T_{Avg}(q) = (3 + 5 + 4 + 5 + 4) / 5 = 4.2$ (time slots). It means that each mobile user has to spend 4.2 time slots averagely on receiving d_1 and d_3 from the broadcast channel.

Example 2. The average access time for multiple query sets

Given a database $D = \{d_1, d_2, d_3, d_4, d_5, d_6\}$, all mobile users issue three query sets in a query profile Q as Table 3.2 shows. There will be 6!/6 possible broadcast programs after scheduling the six data items in D, e.g. $[d_1, d_2, d_3, d_4, d_5, d_6]$, $[d_1, d_2, d_3, d_4, d_6, d_5]$, $[d_1, d_4, d_3, d_2, d_5, d_6]$ …, etc. For the specific broadcast program, $\Phi = [d_3, d_2, d_6, d_4, d_5, d_1]$, as shown in Figure 3.2, we compute the average access time for q_1, q_2 and q_3 as

follows: $T_{Avg}(q_1)$ = (6 + 6 + 6 + 5 + 4 + 3) / 6 = 5 (time slots), $T_{Avg}(q_2)$ = (5 + 6 + 5 + 4 + 6 + 6) / 6 = 5.33 (time slots), and $T_{Avg}(q_3)$ = (3 + 6 + 5 + 6 + 5 + 4) / 6 = 4.83 (time slots). By Equation (3.1), the average access time for the multiple query sets in Q can be computed as $T_{Avg}(Q) = \sum_{j=1}^{3} T_{Avg}(q_j) \times P_j = 5 \times 0.7 + 5.33 \times 0.2 + 4.83 \times 0.1 = 5.049$ (time slots).

Table 3.2. A query profile Q.

Query Set q_j	Probability P_j
$q_1 = \{d_1, d_2, d_3\}$	0.7
$q_2 = \{d_3, d_4, d_5\}$	0.2
$q_3 = \{d_3, d_6\}$	0.1

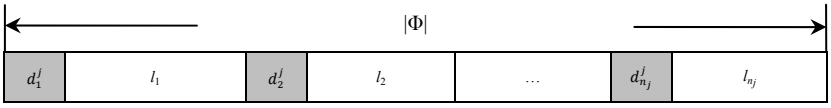
d_3	d_2	d_6	d_4	d_5	d_1

Figure 3.2. A broadcast program Φ.

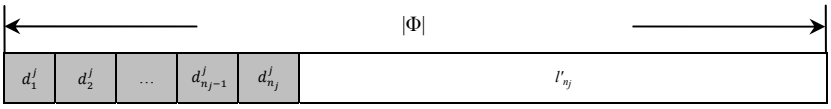
3.2. Coherence degree of a query set

In [12], a measure named *Query Distance* (*QD*) is defined for representing the coherence degree of a query set. The less *QD* of a query set, the better coherence degree it will have. For example, assume that there is a database $D = \{d_1, d_2, \ldots, d_n\}$, all data items in D are of equal size, and a query set $q_j = \left\{d_1^j, d_2^j, \ldots, d_{n_j}^j\right\} \subset D$, $n_j < n$. Without data-item replication, Figure 3.3-a shows that the n data items in D are scheduled arbitrarily in a broadcast program Φ over which the n_j data items in q_j are marked as the dark grids. The n_j segments in Φ, denoted by $l_1, l_2, \ldots, l_{n_j}$, are made up of all data items in D except those in q_j. In contrast, Figure 3.3-b shows that all data items in q_j are clustered together. According to the definition of query distance, the query distance of q_j

in Figure 3.3-a equals |Φ| minus the maximum (length) of the n_j segments. Similarly, that in Figure 3.3-b equals |Φ| – l'_{n_j}. Because each segment in Figure 3.3-a must be shorter than l'_{n_j}, the coherence degree of q_j in Figure 3.3-b is better than that in Figure 3.3-a.



3.3-a. A case of lower coherence degree.



3.3-b. A case of higher coherence degree.

Figure 3.3. Coherence degree of a query set in a broadcast program.

The following Proposition 1 formulates the property of *coherence degree*. The proof of Proposition 1 can be found in [12, 29].

Proposition 1. *A higher coherence degree of a query set q_j causes a lower average access time for retrieving all data items in q_j from a broadcast program without data-item replication in a single channel environment*.

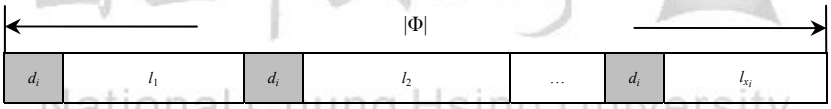
3.3. Optimal scheduling with data-item replication on a single channel

The following Proposition 2 formulates the optimal scheduling with data-item replication in a single channel environment. The proof of Proposition 2 can be found in [9, 14, 15, 18-21, 28, 35, 37].

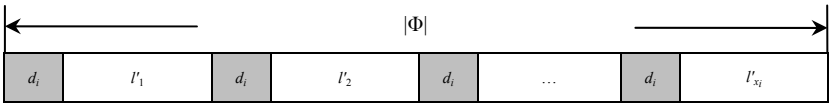
Proposition 2. *An optimal scheduling with data-item replication is achieved when all*

replicas of each data item are equally spaced, *and the ratio of the numbers of each two data items* d_a *and* d_b *satisfies* $\sqrt{p_a/\Delta_a} : \sqrt{p_b/\Delta_b}$, *where* Δ_a *and* Δ_b *represent the size of* d_a *and* d_b, *respectively*.

For example, assume that there is a database $D = \{d_1, d_2, \ldots, d_n\}$ with the corresponding access probabilities, $p_1, p_2, \ldots, p_n$, and the corresponding sizes, $\Delta_1, \Delta_2, \ldots, \Delta_n$. The corresponding numbers of data-item's replicas are $x_1, x_2, \ldots, x_n$, where x_a: x_b are assumed to be equal to $\sqrt{p_a/\Delta_a} : \sqrt{p_b/\Delta_b}$, $1 \le a, b \le n$, $a \ne b$. As shown in Figure 3.4-a, all replicas are scheduled arbitrarily in a broadcast program Φ. The segments between two consecutive replicas of d_i ($1 \le i \le n$) are $\{l_1, l_2, \ldots, l_{x_i}\}$, which are composed of data items in D other than d_i, i.e., $d_1, d_2, \ldots, d_{i-1}, d_{i+1}, \ldots, d_n$. Figure 3.4-b shows the optimal case for retrieving d_i, where $l'_1 = l'_2 = \ldots = l'_{x_i}$.



3.4-a. An arbitrary case.



3.4-b. An optimal case.

Figure 3.4. A broadcast program with data-item replication.

3.4. Optimal partition of data scheduling on multiple channels

The following Proposition 3 formulates the optimal partition of data scheduling without data-item replication for data items of variable size in the multiple channels environment. The proof of Proposition 3 can be found in [11, 16, 38, 39].

Proposition 3. *In the optimal partition of data scheduling for data items of variable size, for any two partitions,* $Group_p$, $Group_q$, *if* $|Group_p| < |Group_q|$, *then* $\forall d_a \in Group_p$, $d_b \in Group_q$, $p_a \geq p_b$. *In addition,* $|Group_{i-1}| \leq |Group_i| \leq \left\lfloor \frac{TL - \sum_{j=1}^{i-1} |Group_j|}{K-i+1} \right\rfloor$, *where* $1 \leq i \leq K$ *and TL denotes the total size of data items.*

For example, assume that a database $D = \{d_1, d_2, \ldots, d_n\}$ consists of n data items with corresponding access probabilities, $p_1, p_2, \ldots, p_n$, $(p_i < p_{i+1}, 1 \leq i \leq n)$ and corresponding sizes, $\Delta_1, \Delta_2, \ldots, \Delta_n$. The broadcast server is equipped with K broadcast channels. First, the n data items are sorted in descending order of their access propabilities. Then, they are partitioned into K groups, and the size of each group is decided according to Proposition 3. The K groups are broadcast on their corresponding channels, $Channel_1, Channel_2, \ldots, Channel_K$. Figure 3.5 shows an optimal partition for scheduling all data items of variable size on K broadcast channels without data-item replication.

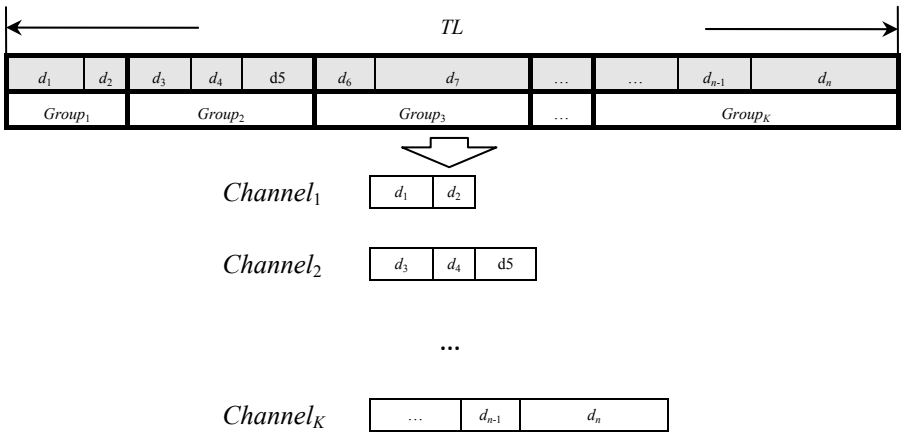


Figure 3.5. Optimal partition for data scheduling on multi-channel.

4 Method

In this section, a scheduling algorithm with a query-set-based concept will be proposed. In section 4.1, a new view of a compound data item for a query set will be presented. Section 4.2 derives a general equation of the optimal average access time for a single query set without data-item replication in a single channel environment. In section 4.3, two general equations are derived for the near-optimal average access time in a broadcast program with query-set replication on multiple channels. Finally, the query-set-based scheduling algorithm and the related algorithms will be proposed in section 4.4.

4.1. View of a *compound data item*

This study is motivated by treating a query set as a compound data item, abbreviated as CDI, by Proposition 1. As in data mining, a set of items is called an *itemset*, and an itemset of size k is referred to a *k-itemset*. Suppose that there are n distinct items. There will be $2^n - 1$ itemsets which are made up of C_1^n 1-itemsets, C_2^n 2-itemsets, …, and C_n^n n-itemsets. If each itemset is viewed as a query set, there are $2^n - 1$ possible user-issued query sets for n data items. Actually, the $2^n - 1$ query sets are unlikely to be issued by mobile users at the same time in real applications.

Consider the query profile Q in Table 3.2 as an example. Six data items, $\{d_1, d_2, d_3, d_4, d_5, d_6\}$, may generate $2^6 - 1$ itemsets (CDIs), including six 1-itemsets, $\{d_1\}$, $\{d_2\}$, …, $\{d_6\}$, fifteen 2-itemsets, $\{d_1, d_2\}$, $\{d_1, d_3\}$, …, $\{d_5, d_6\}$, …, one 6-itemset, $\{d_1, d_2, d_3, d_4, d_5, d_6\}$. If each is viewed as one query set, an extended query profile Q' can be obtained as shown in Table 4.1. Obviously, mobile users access only three of all the CDIs. Concerning query set q_3, the mobile users who issue q_3 need only the CDI, $\{d_3, d_6\}$, not

$\{d_3\}$ nor $\{d_6\}$. Therefore, the Proposition 2 in section 3.3 can be extended to derive a general equation of the near-optimal average access time for multiple query sets (or CDIs) of variable size with query-set replication.

Table 4.1. An extended query profile Q'.

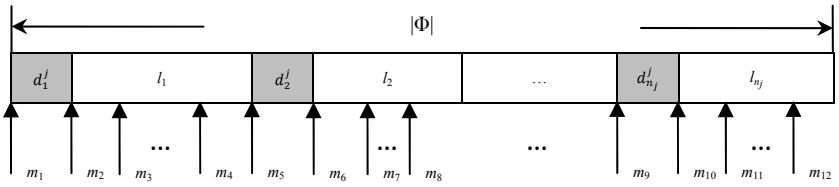
CDI_j	Probability P_j
$\{d_1\}$	0
$\{d_2\}$	0
…	0
$CDI_3 = \{d_3, d_6\}$	0.1
…	0
$CDI_1 = \{d_1, d_2, d_3\}$	0.7
$CDI_2 = \{d_3, d_4, d_5\}$	0.2
…	0
$\{d_1, d_2, d_3, d_4, d_5, d_6\}$	0

4.2. Optimal average access time for a single query set without data-item replication in a single channel

In this section, we derive a general equation of the optimal average access time for retrieving data items in a single query set from a broadcast program without data-item replication as follows.

Without loss of generality, we assume that the database $D = \{d_1, d_2, \ldots, d_n\}$, all data items are equal size, and a query set $q_j = \left\{d_1^j, d_2^j, \ldots, d_{n_j}^j\right\} \subset D$, $n_j < n$. Without data-item replication, Figure 3.3-a shows that the n data items in D are scheduled arbitrarily in a broadcast program Φ over which the n_j data items in q_j are marked as the

dark grids, and the cycle length |Φ| equals n. The n_j segments in Φ, l_1, l_2, …, l_{n_j}, ($0 \le l_i \le |\Phi| - n_j$, $1 \le i \le n_j$) are made up of all data items in D except those in q_j. In contrast, Figure 3.3-b shows the optimal scheduling in QEM where all data items in q_j are cohered together, and $l_1 = l_2 = \ldots = 0$, $l_{n_j} = |\Phi| - n_j$. There are |Φ| possible moments where the mobile users may tune in; therefore, we only choose some of them, m_1, m_2, …, m_{12}, to represent for the sake of brevity.



4.1-a. An arbitrary case.

4.1-b. The optimal case.

Figure 4.1. A broadcast program without data-item replication.

The average access time, $T_{Avg}(q_i)$, for retrieving all data items in q_i from Φ equals the sum of the access time at all possible moments divided by |Φ|. It takes $|\Phi| - l_{n_j}$ time slots on retrieving all data items in q_j at m_1, |Φ| time slots at m_2, $|\Phi| - 1$ time slots at m_3, …, $|\Phi| - l_1 + 1$ time slots at m_4. Therefore, the sum of access time from m_1 to m_4 equals $\frac{2|\Phi|l_1 - l_1^2 + l_1 + 2|\Phi| - 2l_{n_j}}{2}$. Similarly, the sum of that from m_5 to m_8 equals $\frac{2|\Phi|l_2 - l_2^2 + l_2 + 2|\Phi| - 2l_1}{2}$, and the sum of that from m_9 to m_{12} equals $\frac{2|\Phi|l_{n_j} - l_{n_j}^2 + l_{n_j} + 2|\Phi| - 2l_{n_j-1}}{2}$. Hence, the average access time for retrieving all data items in q_j from Φ can be formulated as

$$T_{Avg}(q_j) = \left(\frac{2|\Phi|l_1 - l_1^2 + l_1 + 2|\Phi| - 2l_{n_j}}{2} + \frac{2|\Phi|l_2 - l_2^2 + l_2 + 2|\Phi| - 2l_1}{2} + \cdots + \frac{2|\Phi|l_{n_j} - l_{n_j}^2 + l_{n_j} + 2|\Phi| - 2l_{n_j-1}}{2} \right) \div |\Phi|$$
$$= \frac{2|\Phi| \sum_{i=1}^{n_j} l_i - \sum_{i=1}^{n_j} l_i^2 - \sum_{i=1}^{n_j} l_i + 2n_j|\Phi|}{2|\Phi|}.$$

(4.1)

Because |Φ| and n_j are constants, $\sum_{i=1}^{n_j} l_i$ is also a constant and equal to |Φ| – n_j. Then, Equation (4.1) can be rewritten as

$$T_{Avg}(q_j) = \frac{2(|\Phi| - n_j)|\Phi| + 2n_j|\Phi| - (|\Phi| - n_j) - \sum_{i=1}^{n_j} l_i^2}{2|\Phi|}.$$

(4.2)

Observe that Equation (4.2) can be minimized by maximizing $\sum_{i=1}^{n_j} l_i^2$. The maximum of $\sum_{i=1}^{n_j} l_i^2$ is derived by the following lemma.

Lemma 1. For $\sum_{i=1}^{n_j} l_i$ = |Φ| – n_j, 0 ≤ l_i ≤ |Φ| – n_j, the maximum of $\sum_{i=1}^{n_j} l_i^2$ equals (|Φ| – n_j)^2.

Proof. Let k_1, k_2, …, k_{n_j} be a sequence of integers, where $\sum_{i=1}^{n_j} k_i$ = 2 and 0 ≤ k_i ≤ 2. By the Multinomial theorem [6], ($l_1 + l_2 + \ldots + l_{n_j}$)^2 can be expressed as

$$\left(l_1 + l_2 + \cdots + l_{n_j}\right)^2 = \sum_{k_1, k_2, \ldots, k_{n_j}} \binom{2}{k_1, k_2, \ldots, k_{n_j}} l_1^{k_1} l_2^{k_2} \ldots l_{n_j}^{k_{n_j}}$$
$$= \left(|\Phi| - n_j\right)^2$$

(4.3)

where $\binom{2}{k_1, k_2, \ldots, k_{n_j}} = \frac{2!}{k_1! k_2! \ldots k_{n_j}!}$. Let k'_1, k'_2, …, k'_{n_j} be another sequence of integers, $\sum_{i=1}^{n_j} k'_i$ = 2 and 0 ≤ k'_i ≤ 1. By Equation (4.3), $\sum_{i=1}^{n_j} l_i^2$ can be formulated as

$$\sum_{i=1}^{n_j} l_i^2 = \left(|\Phi| - n_j\right)^2 - \sum_{k_1', k_2', \ldots, k_{n_j}'} \binom{2}{k_1', k_2', \ldots, k_{n_j}'} l_1^{k_1'} l_2^{k_2'} \ldots l_{n_j}^{k_{n_j}'}.$$

(4.4)

Let F be a polynomial function, and

$$F = \sum_{k_1', k_2', \ldots, k_{n_j}'} \binom{2}{k_1', k_2', \ldots, k_{n_j}'} l_1^{k_1'} l_2^{k_2'} \ldots l_{n_j}^{k_{n_j}'} \geq 0.$$

(4.5)

Then, Equation (4.4) can be reformulated as

$$\sum_{i=1}^{n_j} l_i^2 = \left(|\Phi| - n_j\right)^2 - F.$$

(4.6)

By Equation (4.6), there exists a maximum of $\sum_{i=1}^{n_j} l_i^2$ when F equals zero. It is observed from Equation (4.5) that F will equal zero when at most one of l_i does not equal zero. As a result, the maximum of $\sum_{i=1}^{n_j} l_i^2$ equals $(|\Phi| - n_j)^2$. The proof is completed. ■

By Lemma 1, let $l_1 = l_2 = \ldots = l_{n_j-1} = 0$ and $l_{n_j} = |\Phi| - n_j$. That is, all data items of q_j are scheduled coherently as shown in Figure 3.3-b. A significant equation can be derived by replacing the term $\sum_{i=1}^{n_j} l_i^2$ in Equation (4.2) with $(|\Phi| - n_j)^2$ as follows.

$$T_{Avg}(q_j) = \frac{|\Phi|^2 + (2n_j - 1)|\Phi| - (n_j^2 - n_j)}{2|\Phi|}.$$

(4.7)

Equation (4.7) is the general equation of the optimal average access time for retrieving n_j data items in q_j from a broadcast program Φ when all data items in q_j are scheduled coherently in Φ. It is an important component for our scheduling algorithm.

4.3. Near-optimal average access time for multiple query sets with query-set replication in a single channel

In this section, we derive two general equations of the near-optimal average access time for multiple query sets from a broadcast program with query-set replication in a single channel environment. This derivation is motivated by Proposition 2. The only difference, however, is that a CDI is divisible, but a data item is in-divisible. Based on the viewpoint of CDI, the problem of scheduling with query-set replication can be transformed into that of scheduling with data-item (variable size) replication. The near-optimal equation can be used to design a query-set-based scheduling algorithm minimizing average access time.

Without loss of generality, we assume that there is a query profile $Q = \{(q_1, P_1), (q_2, P_2), \ldots, (q_\lambda, P_\lambda)\}$ with the corresponding number of replicas $x_1, x_2, \ldots, x_\lambda$. As shown in Figure 4.1, the broadcast program Φ is made up of all replicas of each query set, and the cycle length $|\Phi|$ equals $\sum_{j=1}^{\lambda} n_j x_j$. The x_j's replicas of query set q_j, marked as the dark grids, are scheduled arbitrarily over Φ. The segments between two consecutive replicas of q_j ($1 \le j \le \lambda$) are $l_1, l_2, \ldots, l_{x_j}$, which are composed of the data items in all query sets except in q_j. There are $|\Phi|$ possible moments where mobile users may tune in, and m_1, m_2, … represent some of them for the sake of illustration. There are two cases of retrieving data items from Φ as shown in Figures 4.2-a and 4.2-b. Figure 4.2-a called *In-divisible*, denotes that a mobile user has to listen until the next coming q_j is totally retrieved by him/her. Figure 4.2-b called *Divisible*, indicates that a mobile user who tunes into Φ at the moment of the middle of broadcasting q_j has to listen until the missed part of q_j is retrieved from the next coming q_j.

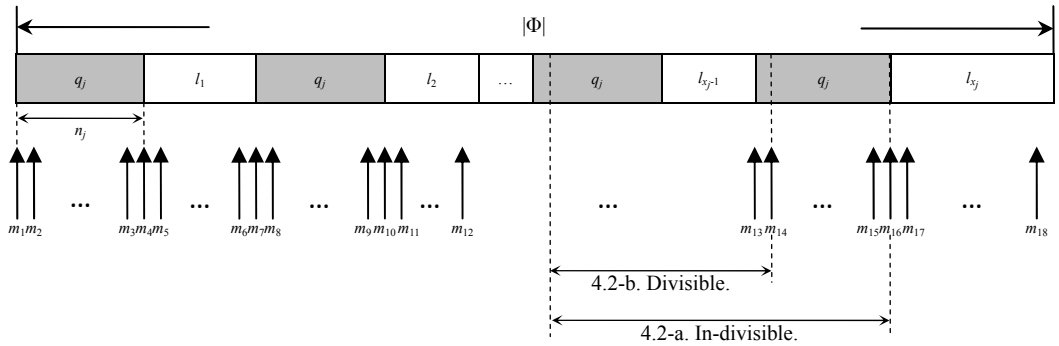


Figure 4.2. A broadcast program with query-set replication.

In the Divisible case, the average access time, $T_{Avg}(q_i)$, for retrieving all data item in q_j from Φ equals the sum of the access time at all possible moments divided by |Φ|. It takes n_j time slots to retrieve all data item in q_j at m_1, i.e., $2n_j + l_1 - 1$ time slots at m_2, …, $n_j + l_1 + 1$ time slots at m_3, $n_j + l_1$ time slots at m_4, $n_j + l_1 - 1$ time slots at m_5, …, $n_j + 1$ time slots at m_6. Therefore, the sum of access time from m_1 to m_6 equals $\frac{l_1^2+(4n_j-1)l_1-n_j+3n_j^2}{2}$. Similarly, the sum of access time from m_7 to m_{12} equals $\frac{l_2^2+(4n_j-1)l_2-n_j+3n_j^2}{2}$. Hence, the average access time for retrieving all data items in q_j can be formulated as

$$T_{Avg}(q_j) = \left(\frac{l_1^2 + (4n_j - 1)l_1 - n_j + 3n_j^2}{2} + \frac{l_2^2 + (4n_j - 1)l_2 - n_j + 3n_j^2}{2} + \cdots + \frac{l_{x_j}^2 + (4n_j - 1)l_{x_j} - n_j + 3n_j^2}{2} \right) \div |\Phi|$$

$$= \frac{\sum_{i=1}^{x_j} l_i^2 + (4n_j - 1)\sum_{i=1}^{x_j} l_i - n_j x_j + 3n_j^2 x_j}{2|\Phi|}. \quad (4.8)$$

Because |Φ|, n_j and x_j are constants, $\sum_{i=1}^{x_j} l_i$ is also a constant and equal to $|\Phi| - n_j x_j$. Then, Equation (4.8) can be rewritten as

$$T_{Avg}(q_j) = \frac{\sum_{i=1}^{x_j} l_i^2 + (4n_j - 1)\sum_{i=1}^{x_j} l_i - n_j x_j + 3n_j^2 x_j}{2|\Phi|}$$
$$= \frac{\sum_{i=1}^{x_j} l_i^2 + (4n_j - 1)(|\Phi| - n_j x_j) - n_j x_j + 3n_j^2 x_j}{2|\Phi|}.$$

(4.9)

Observe that Equation (4.9) can be minimized by minimizing $\sum_{i=1}^{x_j} l_i^2$. The minimum of $\sum_{i=1}^{x_j} l_i^2$ is derived by the following lemma.

Lemma 2. For $\sum_{i=1}^{x_j} l_i = |\Phi| - n_j x_j$, $0 \le l_i \le |\Phi| - n_j x_j$, the minimum of $\sum_{i=1}^{n_j} l_i^2$ equals $\frac{(|\Phi| - n_j x_j)^2}{x_j}$.

Proof. Let $k_1, k_2, \ldots, k_{x_j}$ be a sequence of nonnegative integers, and $\sum_{i=1}^{x_j} k_i = s$ where s is a nonnegative integer. Hence, a simultaneous equation is expressed as

$$\begin{cases} \sum_{i=1}^{x_j} l_i = |\Phi| - n_j x_j \\ \sum_{i=1}^{x_j} k_i = s \end{cases}$$

(4.10)

By Cauchy-Schwarz inequality, Equation (4.10) can be formulated as

$$\left(\sum_{i=1}^{x_j} l_i^2\right)\left(\sum_{i=1}^{x_j} k_i^2\right) \ge \left(\sum_{i=1}^{x_j} l_i k_i\right)^2$$

(4.11)

The left-hand side equals the right-hand side of Equation (4.11) as $l_1 = ck_1$, $l_2 = ck_2$, …, $l_{x_j} = ck_{x_j}$ where c is a constant of real number. Assuming that $k_1 = k_2 = \ldots = k_{n_j} = 1$, Inequality (4.11) can be rewritten as

$$\left(\sum_{i=1}^{x_j} l_i^2\right) \cdot x_j \geq \left(\sum_{i=1}^{x_j} l_i\right)^2$$
$$\Rightarrow \left(\sum_{i=1}^{x_j} l_i^2\right) \geq \frac{\left(|\Phi| - n_j x_j\right)^2}{x_j}.$$

(4.12)

By Equation (4.12), the minimum of $\sum_{i=1}^{x_j} l_i^2$ equals $\frac{\left(|\Phi|-n_j x_j\right)^2}{x_j}$ where $l_1 = l_2 = \ldots = l_{x_j} = \frac{|\Phi|-n_j x_j}{x_j}$. The proof is completed. ■

By Lemma 2, let $l_1 = l_2 = \ldots = l_{x_j}$; that is, all replicas of q_j are equally spaced. An equation can be derived by replacing $\sum_{i=1}^{x_j} l_i^2$ in Equation (4.9) with $\frac{\left(|\Phi|-n_j x_j\right)^2}{x_j}$ as follows.

$$T_{Avg}(q_j) = \frac{|\Phi|^2 + 2n_j x_j |\Phi| - |\Phi| x_j}{2|\Phi| x_j}.$$

(4.13)

By Definition 1, the average access time for all query sets in Q with their corresponding access probabilities can be expressed as

$$T_{Avg}(Q) = \sum_{j=1}^{\lambda} T_{Avg}(q_j) \times P_j$$
$$= \sum_{j=1}^{\lambda} P_j \frac{|\Phi|^2 + 2n_j x_j |\Phi| - |\Phi| x_j}{2|\Phi| x_j}.$$

(4.14)

Equation (4.14) is a function of λ variables and can be optimized by finding an optimal solution of (x_1, x_2, …, x_λ). The following lemma derives the optimal solution of (x_1, x_2, …, x_λ)

Lemma 3. The optimal solution of Equation (4.14) must satisfy that $\frac{x_a}{x_b} = \frac{\sqrt{P_a/n_a}}{\sqrt{P_b/n_b}}$, $1 \leq a, b \leq \lambda$, $a \neq b$.

Proof. Let T be a function of λ variables $x_1, x_2, \ldots, x_\lambda$, and $T(x_1, x_2, \ldots, x_\lambda) = T_{Avg}(Q)$ in Equation (4.14). The partial derivative [27] of T with respect to x_1 can be formulated as

$$\begin{aligned}
\frac{\partial T}{\partial x_1} &= \frac{\partial}{\partial x_1}\left(\sum_{j=1}^{\lambda} P_j \frac{|\Phi|^2 + 2n_j x_j|\Phi| - |\Phi|x_j}{2|\Phi|x_j}\right) \\
&= \frac{\partial}{\partial x_1}\left[\left(\frac{P_1|\Phi|}{2x_1} + \frac{P_2|\Phi|}{2x_2} + \cdots + \frac{P_\lambda|\Phi|}{2x_\lambda}\right) + (P_1 n_1 + P_2 n_2 + \cdots + P_\lambda n_\lambda)\right. \\
&\quad \left. - \left(\frac{P_1}{2} + \frac{P_2}{2} + \cdots + \frac{P_\lambda}{2}\right)\right] \\
&= \frac{\partial}{\partial x_1}\left[\left(\frac{P_1|\Phi|}{2x_1} + \frac{P_2|\Phi|}{2x_2} + \cdots + \frac{P_\lambda|\Phi|}{2x_\lambda}\right) + (P_1 n_1 + P_2 n_2 + \cdots + P_\lambda n_\lambda) - \frac{1}{2}\right] \\
&= \frac{2P_1 n_1 x_1 - 2P_1|\Phi|}{4x_1^2} + \frac{P_2 n_1}{2x_2} + \frac{P_3 n_1}{2x_3} + \cdots + \frac{P_\lambda n_1}{2x_\lambda} \\
&= -\frac{P_1|\Phi|}{2x_1^2} + n_1 \sum_{j=1}^{\lambda} \frac{P_j}{2x_j} = 0 \\
\therefore\ & \frac{P_1|\Phi|}{2n_1 x_1^2} = \sum_{j=1}^{\lambda} \frac{P_j}{2x_j}.
\end{aligned}$$

The partial derivative of T with respect to x_2 can be formulated as

$$\frac{\partial \mathrm{T}}{\partial x_2} = \frac{\partial}{\partial x_2}\left(\sum_{j=1}^{\lambda} P_j \frac{|\Phi|^2 + 2n_j x_j|\Phi| - |\Phi|x_j}{2|\Phi|x_j}\right) = 0$$

$$\therefore \frac{P_2|\Phi|}{2n_2 x_2^2} = \sum_{j=1}^{\lambda} \frac{P_j}{2x_j}.$$

Those of T with respect to x_3 to x_λ can be derived similarly. Thus, the optimal solution can be obtained as

$$\begin{aligned}
&\frac{P_1|\Phi|}{2n_1 x_1^2} = \frac{P_2|\Phi|}{2n_2 x_2^2} = \cdots = \frac{P_\lambda|\Phi|}{2n_\lambda x_\lambda^2} \\
&\therefore \frac{x_a}{x_b} = \frac{\sqrt{P_a/n_a}}{\sqrt{P_b/n_b}}, 1 \le \forall a, b \le \lambda, a \ne b
\end{aligned} \tag{4.15}$$

The proof is completed. ■

By Lemma 3, Equation (4.14) can be reformulated as

$$
\begin{aligned}
T_{Avg}(Q) &= \sum_{j=1}^{\lambda} P_j \frac{|\Phi|^2 + 2n_j x_j |\Phi| - |\Phi| x_j}{2|\Phi| x_j} \\
&= \left[\frac{P_1}{2}\left(n_1\frac{x_1}{x_1} + n_2\frac{x_2}{x_1} + \cdots + n_\lambda\frac{x_\lambda}{x_1}\right) + \frac{P_2}{2}\left(n_1\frac{x_1}{x_1} + n_2\frac{x_2}{x_1} + \cdots + n_\lambda\frac{x_\lambda}{x_1}\right) + \cdots + \right. \\
&\quad \left. \frac{P_\lambda}{2}\left(n_1\frac{x_1}{x_1} + n_2\frac{x_2}{x_1} + \cdots + n_\lambda\frac{x_\lambda}{x_1}\right)\right] + (P_1 n_1 + P_2 n_2 + \cdots + P_\lambda n_\lambda) - \\
&\quad \left(\frac{P_1}{2} + \frac{P_2}{2} + \cdots + \frac{P_\lambda}{2}\right) \\
&= \frac{1}{2}\left[\left(P_1 n_1 + P_1 n_2\frac{\sqrt{P_2 n_1}}{\sqrt{P_1 n_2}} + \cdots + P_1 n_\lambda\frac{\sqrt{P_\lambda n_1}}{\sqrt{P_1 n_\lambda}}\right) + \left(P_2 n_1\frac{\sqrt{P_1 n_2}}{\sqrt{P_2 n_1}} + P_2 n_2 + \right.\right. \\
&\quad \left.\left. \ldots + P_2 n_\lambda\frac{\sqrt{P_\lambda n_2}}{\sqrt{P_2 n_\lambda}}\right) + \cdots + \left(P_\lambda n_1\frac{\sqrt{P_1 n_\lambda}}{\sqrt{P_\lambda n_1}} + P_\lambda n_2\frac{\sqrt{P_2 n_\lambda}}{\sqrt{P_\lambda n_2}} + \cdots + P_\lambda n_\lambda\right)\right] + \\
&\quad (P_1 n_1 + P_2 n_2 + \cdots + P_\lambda n_\lambda) - \frac{1}{2} \\
&= \frac{1}{2}\left[\left(\sqrt{P_1 n_1}\sqrt{P_1 n_1} + \sqrt{P_1 n_1}\sqrt{P_2 n_2} + \cdots + \sqrt{P_1 n_1}\sqrt{P_\lambda n_\lambda}\right) + \right. \\
&\quad \left(\sqrt{P_2 n_2}\sqrt{P_1 n_1} + \sqrt{P_2 n_2}\sqrt{P_2 n_2} + \cdots + \sqrt{P_2 n_2}\sqrt{P_\lambda n_\lambda}\right) + \cdots + \\
&\quad \left.\left(\sqrt{P_\lambda n_\lambda}\sqrt{P_1 n_1} + \sqrt{P_\lambda n_\lambda}\sqrt{P_2 n_2} + \cdots + \sqrt{P_\lambda n_\lambda}\sqrt{P_\lambda n_\lambda}\right)\right] + \\
&\quad (P_1 n_1 + P_2 n_2 + \cdots + P_\lambda n_\lambda) - \frac{1}{2} \\
&= \frac{1}{2}\left(\sqrt{P_1 n_1} + \sqrt{P_2 n_2} + \cdots + \sqrt{P_\lambda n_\lambda}\right)\left(\sqrt{P_1 n_1} + \sqrt{P_2 n_2} + \cdots + \sqrt{P_\lambda n_\lambda}\right) + \\
&\quad (P_1 n_1 + P_2 n_2 + \cdots + P_\lambda n_\lambda) - \frac{1}{2} \\
&= \frac{1}{2}\left(\sum_{j=1}^{\lambda}\sqrt{P_j n_j}\right)^2 + \left(\sum_{j=1}^{\lambda} P_j n_j - \frac{1}{2}\right).
\end{aligned}
\tag{4.16}
$$

Equation (4.16) is the general equation of the near-optimal average access time for multiple query sets with query-set replication in a single channel environment.

The following is the derivation of the average access time, $T_{Avg}(q_i)$, for retrieving all data item in q_j from Φ in the In-divisible case. It takes n_j time slots to retrieve all data item in q_j at m_1, i.e., $n_j + l_1$ time slots at m_2, …, $n_j + l_1$ time slots at m_3, $n_j + l_1$ time slots at m_4, $n_j + l_1 - 1$ time slots at m_5, …, $n_j + 1$ time slots at m_6. Therefore, the sum of access time from m_1 to m_6 equals $\frac{l_1^2+(4n_j-1)l_1+2n_j^2}{2}$. Similarly, the sum of access time from m_7

to m_{12} equals $\frac{l_2^2+(4n_j-1)l_2+2n_j^2}{2}$. Hence, the average access time for retrieving all data items in q_j can be formulated as

$$T_{Avg}(q_j) = \left(\frac{l_1^2 + (4n_j - 1)l_1 + 2n_j^2}{2} + \frac{l_2^2 + (4n_j - 1)l_2 + 2n_j^2}{2} + \cdots + \frac{l_{x_j}^2 + (4n_j - 1)l_{x_j} + 2n_j^2}{2}\right) \div |\Phi| = \frac{\sum_{i=1}^{x_j} l_i^2 + (4n_j - 1)\sum_{i=1}^{x_j} l_i + 2n_j^2 x_j}{2|\Phi|}. \quad (4.17)$$

Similar to Equation (4.8), Equation (4.17) can be rewritten by replacing $\sum_{i=1}^{x_j} l_i$ with $|\Phi| - n_j x_j$ as

$$T_{Avg}(q_j) = \frac{\sum_{i=1}^{x_j} l_i^2 + (4n_j - 1)\sum_{i=1}^{x_j} l_i + 2n_j^2 x_j}{2|\Phi|} = \frac{\sum_{i=1}^{x_j} l_i^2 + (4n_j - 1)(|\Phi| - n_j x_j) + 2n_j^2 x_j}{2|\Phi|}. \quad (4.18)$$

By Lemma 2, let $l_1 = l_2 = \ldots = l_{x_j}$; that is, all replicas of q_j are equally spaced. An equation can be derived by replacing $\sum_{i=1}^{x_j} l_i^2$ in Equation (4.18) with $\frac{(|\Phi| - n_j x_j)^2}{x_j}$ as follows.

$$T_{Avg}(q_j) = \frac{|\Phi|^2 + (2n_j - 1)x_j|\Phi| - (n_j^2 - n_j)x_j^2}{2|\Phi|x_j}. \quad (4.19)$$

By Definition 1, the average access time for all query sets in Q with their corresponding access probabilities can be expressed as

$$T_{Avg}(Q) = \sum_{j=1}^{\lambda} T_{Avg}(q_j) \times P_j = \sum_{j=1}^{\lambda} P_j \frac{|\Phi|^2 + (2n_j - 1)x_j|\Phi| - (n_j^2 - n_j)x_j^2}{2|\Phi|x_j}.$$

(4.20)

By Lemma 3, Equation (4.20) can be reformulated as

$$
\begin{aligned}
T_{Avg}(Q) &= \sum_{j=1}^{\lambda} P_j \frac{|\Phi|^2 + (2n_j - 1)x_j|\Phi| - (n_j^2 - n_j)x_j^2}{2|\Phi|x_j} \\
&= \left(\frac{P_1|\Phi|}{2x_1} + \frac{P_2|\Phi|}{2x_2} + \cdots + \frac{P_\lambda|\Phi|}{2x_\lambda}\right) + \\
&\quad \left(\frac{P_1(2n_1 - 1)}{2} + \frac{P_2(2n_2 - 1)}{2} + \cdots + \frac{P_\lambda(2n_\lambda - 1)}{2}\right) - \\
&\quad \left(\frac{P_1 n_1(n_1 - 1)x_1}{2|\Phi|} + \frac{P_2 n_2(n_2 - 1)x_2}{2|\Phi|} + \cdots + \frac{P_\lambda n_\lambda(n_\lambda - 1)x_\lambda}{2|\Phi|}\right) \\
&= \frac{1}{2}\left(\sqrt{P_1 n_1} + \sqrt{P_2 n_2} + \cdots + \sqrt{P_\lambda n_\lambda}\right)^2 + \left((P_1 n_1 + P_2 n_2 + \cdots + P_\lambda n_\lambda) - \frac{1}{2}\right) \\
&\quad - \frac{P_1\sqrt{P_1 n_1}(n_1 - 1) + P_2\sqrt{P_2 n_2}(n_2 - 1) + \cdots + P_\lambda\sqrt{P_\lambda n_\lambda}(n_\lambda - 1)}{2\left(\sqrt{P_1 n_1} + \sqrt{P_2 n_2} + \cdots + \sqrt{P_\lambda n_\lambda}\right)} \\
&= \frac{1}{2}\left(\sum_{j=1}^{\lambda} \sqrt{P_j n_j}\right)^2 + \left(\sum_{j=1}^{\lambda} P_j n_j - \frac{1}{2}\right) - \frac{\sum_{j=1}^{\lambda} \sqrt{P_j n_j}(n_j - 1)}{2\sum_{j=1}^{\lambda} \sqrt{P_j n_j}}.
\end{aligned}
$$

(4.21)

Equation (4.21) is a modification of Equation (4.16). Both of them are employed in deciding the lower bounds of the average access time.

A broadcast program with data-item replication corresponding to Proposition 2 is hard to generate. For example, given three data items of variable size, $\{d_1, d_2, d_3\}$, and their corresponding access probabilities: 1/2, 1/3, 1/6, it denotes that a broadcast program has to transmit d_1 every two time slots, transmit d_2 every three time slots and transmit d_3 every six time slots on a single channel. However, it is impossible to generate such a broadcast program. As a result, approximate scheduling algorithms are proposed in the next section.

4.4. Broadcast program scheduling algorithms

In this section, we present algorithms for generating a near-optimal broadcast

program in multi-channel environments. In section 4.4.1, an algorithm iPBA which is an improvement of algorithm PBA [25] is proposed. Section 4.4.2 presents an algorithm CBS with the concept of Proposition 1. In section 4.4.3, we transform the concept of Proposition 3 into a query-set-based scheduling algorithm.

4.4.1 Improved placement-based allocation algorithm

In [23, 25], an algorithm named PBA, standing for Placement-Based Allocation, is proposed. PBA can generate a broadcast program from the view of the placement and avoid the existence of conflicting data items. Algorithm PBA, however, considers merely that a query set with higher access probability deserves a higher priority to be scheduled.

Lemma 3 indicates that the priority of a query set is proportional to the square root of its access probability and inversely proportional to the square root of its size. Therefore, the consideration of scheduling priority in algorithm PBA is modified in algorithm iPAB by adding the factor of each query set size. The only difference between iPBA and PBA is that we sort the query sets in a query profile according to the square root of the ratio of its access probability to its query set size (as shown in Figure 4.2).

Function Sort(Q) /* Q is a query profile */
BEGIN
1. **for** each two query sets q_a and $q_b \in Q$
2. **do if** $\sqrt{P_a/n_a} < \sqrt{P_b/n_b}$
3. **then** exchange $q_a \leftrightarrow q_b$ /* exchange the context of q_a and q_b */
4. **return** Q

END

Figure 4.3. Function Sort (Q) of algorithm iPBA.

4.4.2 Coherence-based scheduling algorithm

In this section, we propose the second algorithm named CBS, the abbreviation of

Coherence-Based Scheduling algorithm. By Lemma 2, the average access time for retrieving all data items in a query set can be minimized as they are scheduled coherently in a broadcast program (without data-item replication). In the case of multiple query sets, however, it is difficult to generate such a broadcast program for each query set in a single channel every time. For example, given three query sets, $q_1 = \{d_1, d_2, d_3\}$, $q_2 = \{d_2, d_4\}$ and $q_3 = \{d_2, d_5, d_6\}$, d_2 exists in the three query sets; that is, we can only cluster all data items in two of the three query sets together while the d_2 in the third one is left un-clustered. The Coherence-Based Scheduling algorithm, as shown in Figure 4.4, generates a broadcast program without data-item replication in multiple channels. It considers not only the square root of the ratio of a query set's access probability to its query set size, but also the better coherence degree for all query sets. Figure 4.4 briefly shows that all query sets, marked as the dark grids, are partitioned into K groups of identical length. Then, each group is broadcasted cyclically in corresponding broadcast channels.

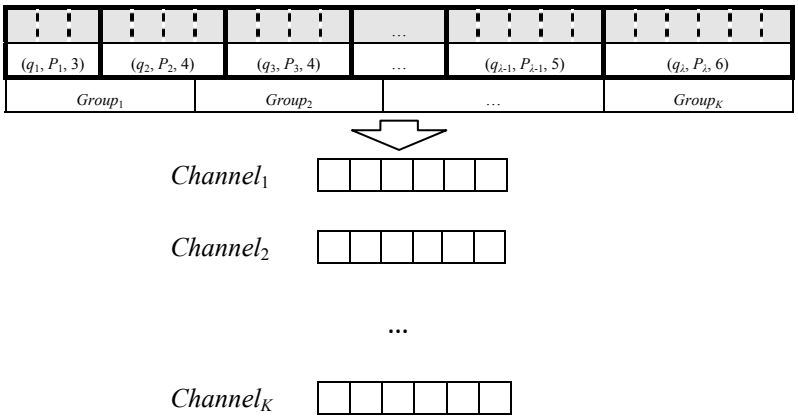


Figure 4.4. Model of CBS.

Algorithm CBF first sorts all query sets in Q by the sorting function shown in Figure 4.3. After that, all data items in each query set are coherently arranged into a 1-dimensional array, named *QSList*, according to the query set's priority. Then, all

replicas of each data item are removed from *QSList*. Lines 13-19 make each data item d_i exist only in the query set with highest priority and remove other replicas of d_i from the query sets with lower priority in *QSList*. Therefore, each data item appears exactly once, and the length of *QSList* is exactly equal to n. Finally, lines 21-25 allocate all data items in *QSList* to the broadcast program *BP* which is made up of K channels with length $\left\lceil \frac{n}{K} \right\rceil$.

```
Algorithm CBS(Q, D, K)  /* a query profile Q, a database D, a number of channels K */
BEGIN
1.  n ← the number of data items in D.
2.  QSL ← the sum of all sizes of query sets in Q.
3.  Create an empty broadcast program BP[1..K][1..⌈n/K⌉].
4.  Create an empty array QSList[1..QSL].
5.  Q ← Sort(Q)  /* Function Sort(Q) is showed in Figure 4.2.*/
6.  /* Copy all data items of each query set in Q to QSList sequentially */
7.  QSL_ptr ← 1
8.  for each q_j ∈ Q
9.      do for i ← 1 to Size(q_j)  /* Function Size(q_j) returns the size of q_j */
10.         do QSList[QSL_ptr] ← q_j[i]
11.        QSL_ptr ← QSL_ptr + 1
12. /* Remove all replicas of each data item */
13. for i ← 1 to QSL
14.     do for j ← i + 1 to QSL
15.         do if QSList[i] = QSList[j]
16.                 then for k ← j + 1 to QSL
17.                         do QSList[k − 1] = QSList[k]
18.                    QSL ← QSL − 1
19.                    j ← i
20. /* Allocate all data items in QSList to BP */
21. QSL_ptr ← 1
22. for i ← 1 to K
23.     do for j ← 1 to ⌈n/K⌉
24.         do BP[i][j] ← QSList[QSL_ptr]
25.            QSL_ptr ← QSL_ptr + 1
26. return BP
END
```

Figure 4.5. Algorithm CBS.

4.4.3 Query-set-based scheduling algorithm

This section proposes a broadcast program scheduling algorithm named QSBS, the abbreviation of Query-Set-Based Scheduling algorithm, based on Propositions 1, 2, and

3, in the multi-channel environment. It is adapted from the extension of the heuristic algorithm in [16].

The heuristic algorithm in [16] is proposed to generate a broadcast program for all data items of equal size in multi-channel environments, and its extension is designed theoretically to deal with the data items of variable size. The extension approach is not really implemented in [16], because there is a main problem in it. That is, a data item may be partitioned into two small data items according to the rules in Proposition 3. However, the extension approach is quite adequate for generating a broadcast program for CDIs. A CDI is allowed to be partitioned into two smaller CDIs in algorithm QSBS. Given a query profile $Q = \{(q_1, P_1), (q_2, P_2), \ldots, (q_\lambda, P_\lambda)\}$ and K broadcast channels, Figure 4.6 briefly shows that all query sets, marked as the dark grids, are partitioned into K groups according to the Proposition 3. Then, each group is broadcast cyclically in corresponding broadcast channels.

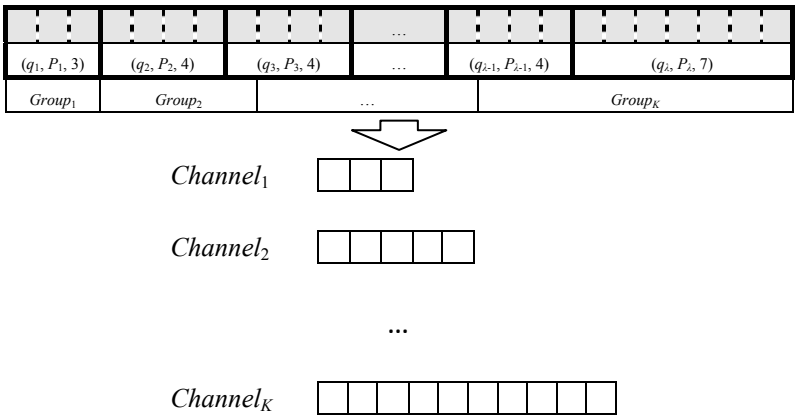


Figure 4.6. Model of QSBS.

Figure 4.7 shows the detail of algorithm QSBS. Line 3 creates a broadcast program of K empty channels with unknown cycle length. Line 5 sorts all query sets according to the sorting function as shown in Figure 4.3. Then, lines 8-11 arrange all query sets into a 1-dimensional array named *QSList*, according to its priority. The ***for-loop*** in lines 14-20

decides the optimal size of each group and the context of each broadcast channel. According to Proposition 3, line 15 firstly computes all the possible $|Group_i|$, $1 \le i \le K$; that is, $|Group_{i-1}|$, $|Group_{i-1}| + 1$, …, $\left\lfloor \frac{TL-\sum_{j=1}^{i-1}|Group_j|}{K-i+1} \right\rfloor - 1$, $\left\lfloor \frac{TL-\sum_{j=1}^{i-1}|Group_j|}{K-i+1} \right\rfloor$. For each possible size, line 16 decides an optimal size of $Group_i$ according to three evaluations as follows.

Evaluation 1. Evaluate the real average access time of retrieving all scheduled query sets from 1st channel to $(i - 1)$-th channel.

For all possible $|Group_i|$, the results of Evaluation 1 are identical, they are set to zero as i equals 1.

Evaluation 2. Evaluate the real average access time of retrieving all query sets from each possible $Group_i$ produced in line 15.

All data items of a query set are clustered together in each possible $Group_i$, and the average access time for $Group_i$ can be computed by combining Definition 1 with Equation (4.7) as follows.

$$T_{Avg}(Group_i) = \sum_{j=1}^{\lambda'} P_j \left(\frac{|\Phi'|^2 - n'^2_j + 2|\Phi'|n'_j - |\Phi'| + n'_j}{2|\Phi'|} \right). \quad (4.22)$$

There will be $\left(\left\lfloor \frac{TL-\sum_{j=1}^{i-1}|Group_j|}{K-i+1} \right\rfloor - |Group_{i-1}| + 1\right)$ distinct values of $T_{Avg}(Group_i)$ produced by Evaluation 2. In Equation (4.22), λ' represents the number of query sets in $Group_i$, and $|\Phi'|$ denotes the $|Group_i|$. Note that $|Group_1|$ must be greater than or equal to the size of the first query set scheduled in *QSList*. A query set may be divided into two

smaller query sets, one is scheduled into the *i*-th channel and the other is scheduled into the next channel. Therefore, n' represents the new size of each query set scheduled into the *i*-th channel.

Evaluation 3. For each possible $|Group_i|$, evaluate the theoretical minimum of access time for the remainder query sets which are optimally scheduled into the other $(K - i)$ groups.

In Evaluation 3, the remainder query sets are assumed to be uniformly scheduled into the other $(K - i)$ groups. Therefore, the theoretical minimum of access time, $T_{Avg}(Group_{i+1 \text{ to } K})$, for the remainder query sets from $Group_{i+1}$ to $Group_K$ can be computed by Equation (4.16) as follows.

$$T_{Avg}(Group_{i+1 \text{ to } K}) = \frac{1}{2(K-i)}\left(\sum_{j=1}^{\lambda'}\sqrt{P'_j n'_j}\right)^2 + \left(\sum_{j=1}^{\lambda'} P'_j n'_j - \frac{1}{2}\right). \quad (4.23)$$

In Equation (4.23), λ' represents the number of the remainder query sets, and n'_j denotes the size of them. P'_j is the corresponding access probability of each remainder query set, and equals P_j divided by the sum of the access probabilities of remainder query sets, where P_j is the initial access probability.

Summing up the results of Evaluations 1, 2 and 3 for each possible $|Group_i|$, lines 16-17 then choose the optimal $|Group_i|$ among all evaluations and assign $|Group_i|$ to BPL_i as the cycle length in the *i*-th channel. Lines 18-20 allocate BPL_i data items in *QSList* to the *i*-th channel of *BP*. Finally, all replicas of each data item are removed from each broadcast channel. Lines 22-29 make each data item d_i exist only in the query set with highest priority and remove other replicas of d_i from the query sets with lower

priority in each channel of *BP*. Concerning the data items among all broadcast channels, a data item d_i is permitted to exist in more than one channel simultaneously; however, there is only one d_i in a channel. Thus, a mobile user does not switch among channels frequently. Assume that there is a query profile $Q = \{(q_1, P_1), (q_2, P_2), \ldots, (q_\lambda, P_\lambda)\}$ with the corresponding query set sizes $n_1, n_2, \ldots, n_\lambda$, and $\sqrt{P_1/n_1} \geq \sqrt{P_2/n_2} \geq \ldots \geq \sqrt{P_\lambda/n_\lambda}$. Let query sets $q_1, q_2, \ldots, q_{j-1}$, and the first ε data items in q_j have been partitioned into $group_1, group_2, \ldots, group_{i-1}$. If the number of the remainder data items in q_j, $n_j - \varepsilon$, is greater than $|Group_i|$, the query set q_j will be partitioned into more than two groups. The condition is derived by the following lemma.

Lemma 4. Let a query set $Q = \{(q_1, P_1), (q_2, P_2), \ldots, (q_\lambda, P_\lambda)\}$ with the corresponding query set sizes $n_1, n_2, \ldots, n_\lambda$, where $\sqrt{P_1/n_1} \geq \sqrt{P_2/n_2} \geq \ldots \geq \sqrt{P_\lambda/n_\lambda}$. If the first ε ($1 \leq \varepsilon \leq n_j$) data items in the query set q_j ($1 \leq j \leq \lambda$) have been partitioned into $group_1$, $group_2, \ldots, group_{i-1}$, the remainder data items in q_j will be partitioned into more than two groups as $n_j - \varepsilon > \frac{n_{j+1}+n_{j+2}+\ldots+n_\lambda}{K-i}$.

Proof. Let *TL* denote the sum of the sizes of all query sets in *Q*. By Proposition 3, the range of $|Group_i|$ can be obtained from

$$|Group_{i-1}| \leq |Group_i| \leq \left\lfloor \frac{TL - \sum_{j=1}^{i-1} |Group_j|}{K-i+1} \right\rfloor$$
$$\Rightarrow |Group_{i-1}| \leq |Group_i| \leq \left\lfloor \frac{n_j - \varepsilon + n_{j+1} + n_{j+2} + \ldots + n_\lambda}{K-i+1} \right\rfloor. \quad (4.24)$$

If we divide q_j into more than two groups, the remainder of q_j must satisfy

$$|Group_i| \leq \left\lfloor \frac{n_j - \varepsilon + n_{j+1} + n_{j+2} + \ \ldots + n_\lambda}{K - i + 1} \right\rfloor < n_j - \varepsilon$$
$$\Rightarrow n_j - \varepsilon > \frac{n_j - \varepsilon + n_{j+1} + n_{j+2} + \ \ldots + n_\lambda}{K - i + 1}$$
$$\Rightarrow n_j - \varepsilon > \frac{n_{j+1} + n_{j+2} + \ \ldots + n_\lambda}{K - i}.$$

(4.25)

The proof is completed. ■

Algorithm QSBS(Q, D, K) /* a query profile Q, a database D, a number of channels K */
BEGIN
1. $n \leftarrow$ the number of data items in D.
2. $QSL \leftarrow$ the sum of all sizes of query sets in Q.
3. Create an empty broadcast program $BP[1..K][unknown]$.
4. Create an empty array $QSList[1..QSL]$.
5. $Q \leftarrow$ Sort(Q) /* Function Sort(Q) is showed in Figure 4.3.*/
6. /* Copy all data items of each query set in Q to $QSList$ sequentially */
7. $QSL_{ptr} \leftarrow 1$
8. **for** each $q_j \in Q$
9. **do for** $i \leftarrow 1$ **to** Size(q_j) /* Function Size(q_j) returns the size of q_j */
10. **do** $QSList[QSL_{ptr}] \leftarrow q_j[i]$
11. $QSL_{ptr} \leftarrow QSL_{ptr} + 1$
12. /* Produce the optimal broadcast program in each channel */
13. $QSL_{ptr} \leftarrow 1$
14. **for** $i \leftarrow 1$ **to** K
15. **do** Compute the possible range of length for $BP[i]$
16. $BPL_i \leftarrow$ the length in the possible range minimizes average access time
17. Create $BP[i][1..BPL_i]$
18. **for** $j \leftarrow 1$ **to** BPL_i
19. **do** $BP[i][j] \leftarrow QSList[QSL_{ptr}]$
20. $QSL_{ptr} \leftarrow QSL_{ptr} + 1$
21. /* Remove all replicas of each data item in each channel */
22. **for** $i \leftarrow 1$ **to** K
23. **do for** $j \leftarrow 1$ **to** BPL_i
24. **do for** $k \leftarrow j + 1$ **to** BPL_i
25. **do if** $BP[i][j] = BP[i][k]$
26. **then for** $m \leftarrow k + 1$ **to** BPL_i
27. **do** $BP[i][m-1] = BP[i][m]$
28. $BPL_i \leftarrow BPL_i - 1$
29. $k = j$
30. **return** BP
END

Figure 4.7. Algorithm QSBS.

5 Experiments

To evaluate the performance (average access time specifically) of the algorithms iPAB, CBS and QSBS, we conduct several simulation experiments and compare them with algorithm PAB [25]. All algorithms have been implemented in Microsoft Visual C# 2005, and simulations are performed on an Intel Core 2 Duo 2.66 GHz machine with 2 Gbytes of RAM running Microsoft Windows Vista Enterprise Edition Service Pack 1.

Table 5.1 lists the default settings and ranges of the parameters in the simulations. As observed from [25], the number of query sets is a minor factor affecting performance. Therefore, we fix the number of query sets λ to be 128. The corresponding access probabilities of all query sets are generated by Zipf distribution [16, 18, 23] with skewness parameter θ. The size of each query set is randomly produced under Poisson distribution and Normal distribution, respectively. Furthermore, the size of each query set has to be greater than or equal to N, the minimal size of each query set. Because each possible size of query set can be paired with any access probability, each simulation is performed 128 times for all possible combinations to ensure the accuracy. In order to investigate the effect of query set size, a ratio of the sum of query set sizes to the number of data items is defined as $\gamma = \frac{\sum_{j=1}^{\lambda} n_j}{n}$. The increase of query set sizes sometimes implies the increase of data items. According to the variation of γ, we can observe the relationship between average access time and average size of all query sets under a constant number of data items n. In the simulations, we make the following two assumptions: (1) the average access time is measured in time slots, and (2) the mobile users have a priori knowledge about the allocation of their interested data items (such as by the index) so that they can tune into the proper channels immediately. All data items of a query set may exist in more than one channel simultaneously; therefore, the average

access time for a query set is defined as the accumulation of the average access time for receiving part of data items in a query set in each channel.

Table 5.1. Simulation Parameters.

Parameters	Default value	Ranges	Descriptions
λ	128	128	The number of query sets in Q.
n	1024	1024 ~ 1824	The number of data items in D.
K	8	4 ~ 64	The number of broadcast channels.
θ	1.0	0.4 ~ 1.6	The parameter of Zipf distribution.
N	15	5 ~ 25	The minimum of query set size.
γ	3.3	2.0 ~ 10.0	The ratio of query set size.

5.1. Comparison between the In-divisible and Divisible cases

In this section, a series of simulations show that whether the average access time in the equation of the Divisible case in section 4.2 is better than that of the In-divisible case. Since the two cases are used to indicate the lower bounds of average access time, the simulations also show the rationality of them. The y-axis of each figure below represents the average access time, and the x-axis shows the value of various parameters. The variation and default value of each parameter are shown in Table 5.1.

The simulations in Figures 5.1 and 5.2 show the effect of the minimal query set size with Normal and Poisson size distributions on the average access time for the Divisible and In-divisible cases. The minimum of query set size N is limited from 5 to 25, and other parameters are set to the default values as shown in Table 5.1. Since the size of each query set is greater than or equal to N, it has to take more access time to retrieve each query set as N increases. As a result, the average access time increases as

N increases.

The simulations in Figures 5.3 and 5.4 represent the effect of the number of data items with Normal and Poisson size distributions on the average access time for the Divisible and In-divisible cases. The number of data items is limited from 1024 to 1824, and other parameters are set to the default values as shown in Table 5.1. Figures 5.3 and 5.4 indicate that the average access time increases as *n* increases. Since the increase of the data item number causes the longer cycle length on each broadcast channel, it will take more access time to retrieve a data item in each query set.

The simulations in Figures 5.5 and 5.6 show the effect of the number of channels with Normal and Poisson size distributions on the average access time for the Divisible and In-divisible cases. The number of channels *K* is limited from 4 to 64, and other parameters are set to the default values as shown in Table 5.1. The two figures illustrate that the average access time decreases 50% approximately as *K* is doubly increased. The reason is that the increase of the channel number decreases the cycle length on each channel. Therefore, it takes less average access time to retrieve each data item from the channel with shorter cycle length.

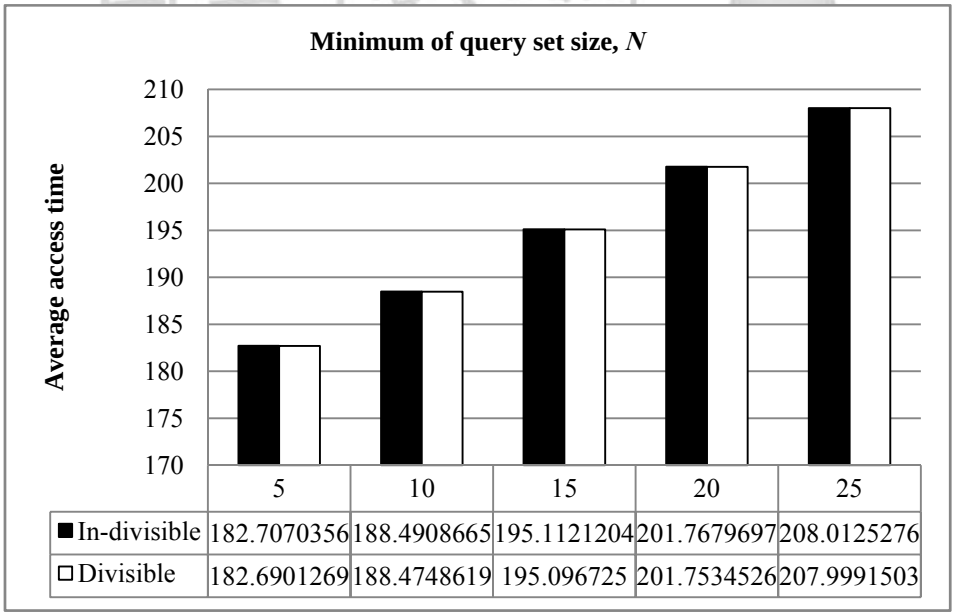
The simulations in Figures 5.7 and 5.8 represent the effect of the skewness with Normal and Poisson size distributions on the average access time for the Divisible and In-divisible cases. The range of skewness θ is limited from 0.4 to 1.6, and other parameters are set to the default values as shown in Table 5.1. The average access time decreases when the skewness θ increases in the figures. As θ increases, the access probabilities are increasingly skewed. A high skew access probabilities means that a small number of query sets are accessed frequently. This explains the average access time shown in Figures 5.7 and 5.8.

The simulations in Figures 5.9 and 5.10 show the effect of the ratio of query set

size to the number of data items with Normal and Poisson size distributions on the average access time for the Divisible and In-divisible cases. The value of γ is limited from 2.0 to 10.0, and other parameters are set to the default values as shown in Table 5.1. The figures illustrate that the average access time increases as γ increases. Under a fixed number of data items n, the increasing of γ indicates the average increasing of all query set sizes. It will take more average access time to retrieve a larger query set.

Overall, the Divisible case shows a slight improvement over the In-divisible case, and both of their curve trends are nearly identical in each pair of simulations with both distributions of query set sizes. There is no anomaly in each simulation; therefore, they can be used as the lower bounds of average access time.

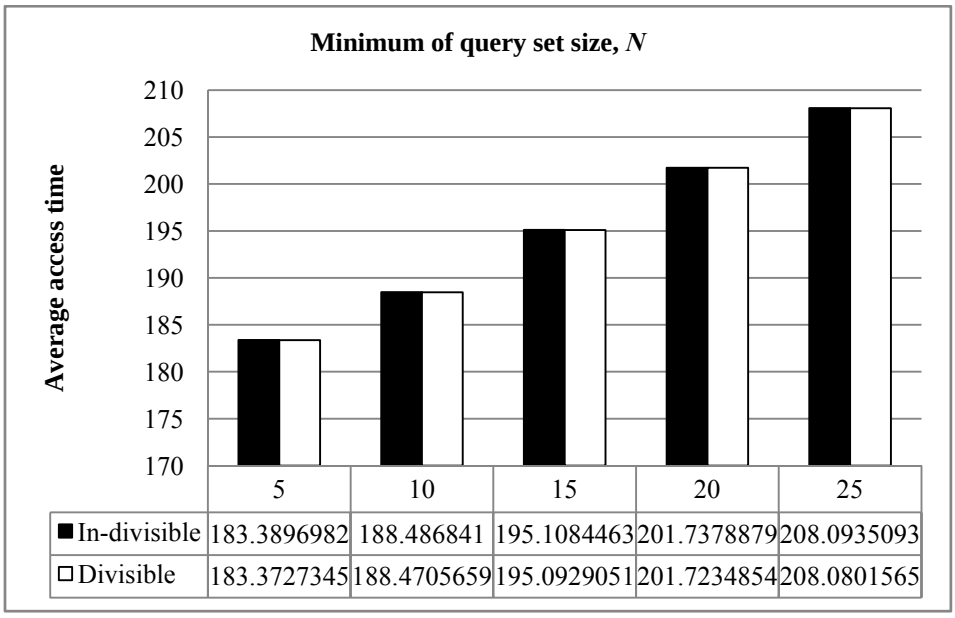
Constant parameters: $\lambda = 128$; $n = 1024$; $K = 8$; $\theta = 1.0$; $\gamma = 3.3$



	5	10	15	20	25
■ In-divisible	182.7070356	188.4908665	195.1121204	201.7679697	208.0125276
□ Divisible	182.6901269	188.4748619	195.096725	201.7534526	207.9991503

Figure 5.1. Effect of the minimal query set size with Normal size distribution.

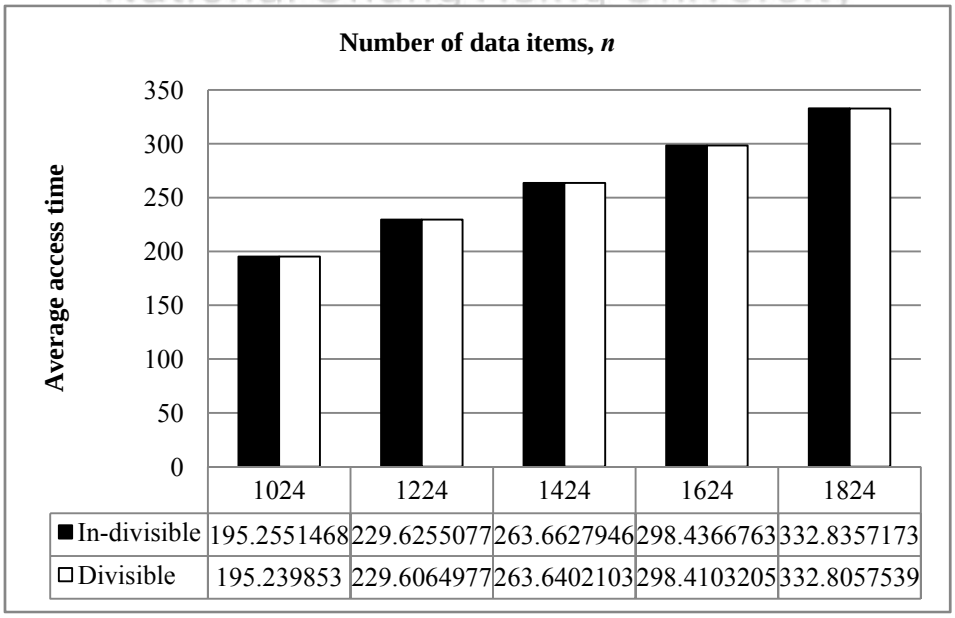
Constant parameters: $\lambda = 128$; $n = 1024$; $K = 8$; $\theta = 1.0$; $\gamma = 3.3$



	5	10	15	20	25
■ In-divisible	183.3896982	188.486841	195.1084463	201.7378879	208.0935093
□ Divisible	183.3727345	188.4705659	195.0929051	201.7234854	208.0801565

Figure 5.2. Effect of the minimal query set size with Poisson size distribution.

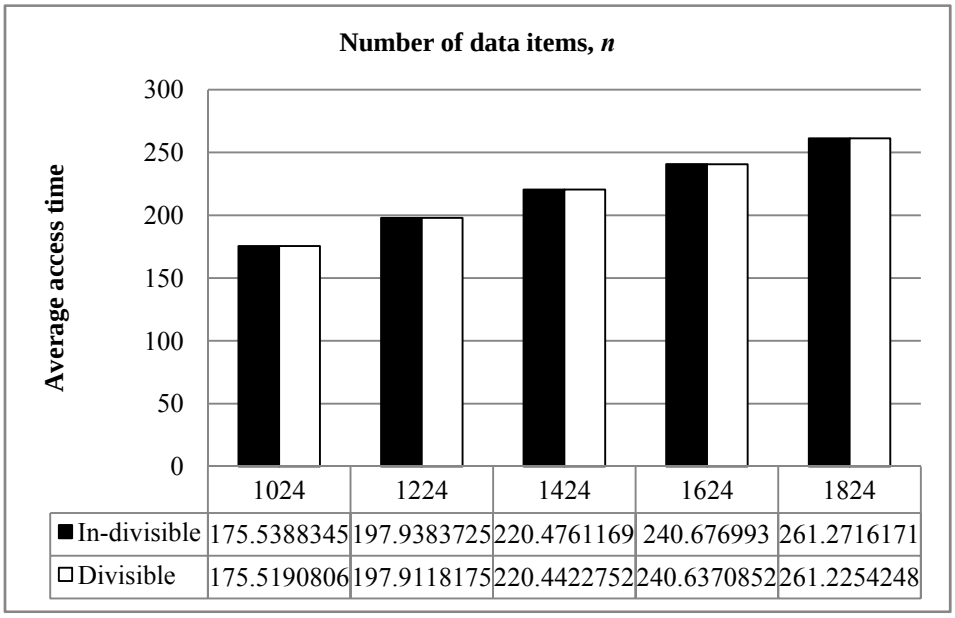
Constant parameters: $\lambda = 128$; $K = 8$; $\theta = 1.0$; $N = 15$; $\gamma = 3.3$



	1024	1224	1424	1624	1824
■ In-divisible	195.2551468	229.6255077	263.6627946	298.4366763	332.8357173
□ Divisible	195.239853	229.6064977	263.6402103	298.4103205	332.8057539

Figure 5.3. Effect of the number of data items with Normal size distribution.

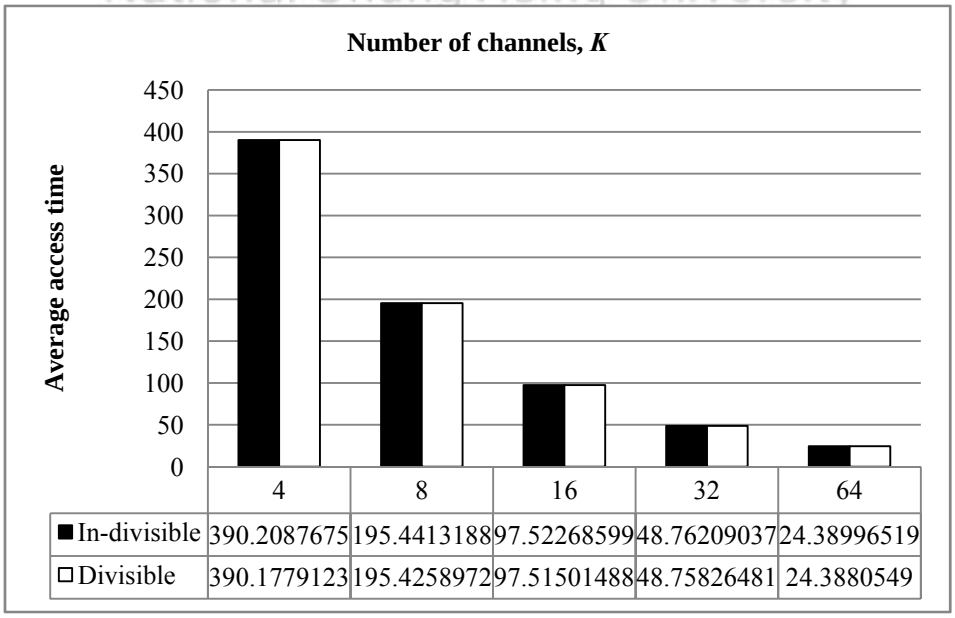
Constant parameters: $\lambda = 128$; $K = 8$; $\theta = 1.0$; $N = 15$; $\gamma = 3.3$



	1024	1224	1424	1624	1824
■ In-divisible	175.5388345	197.9383725	220.4761169	240.676993	261.2716171
□ Divisible	175.5190806	197.9118175	220.4422752	240.6370852	261.2254248

Figure 5.4. Effect of the number of data items with Poisson size distribution.

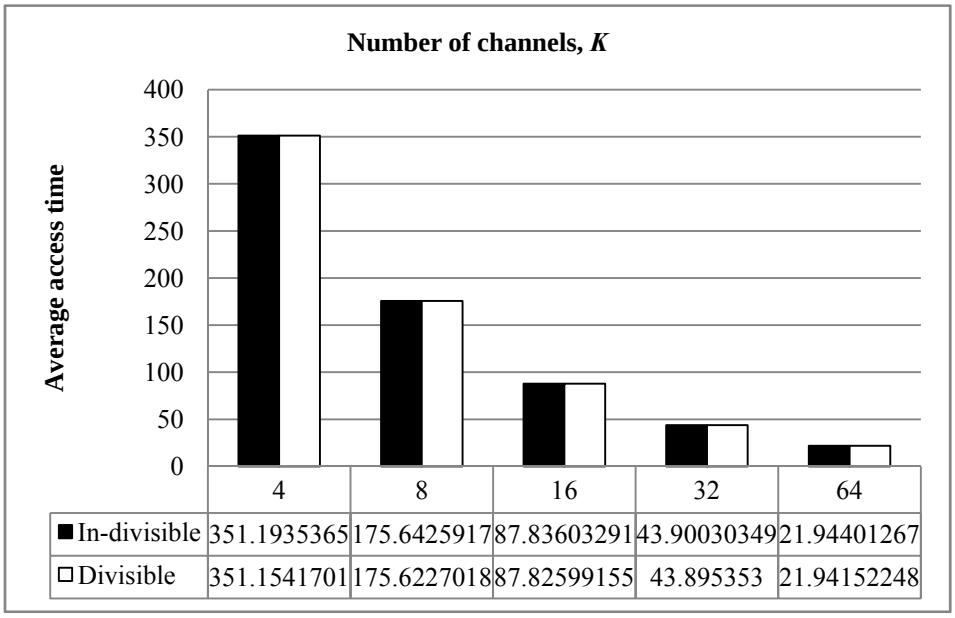
Constant parameters: $\lambda = 128$; $n = 1024$; $\theta = 1.0$; $N = 15$; $\gamma = 3.3$



	4	8	16	32	64
■ In-divisible	390.2087675	195.4413188	97.52268599	48.76209037	24.38996519
□ Divisible	390.1779123	195.4258972	97.51501488	48.75826481	24.3880549

Figure 5.5. Effect of the number of channels with Normal size distribution.

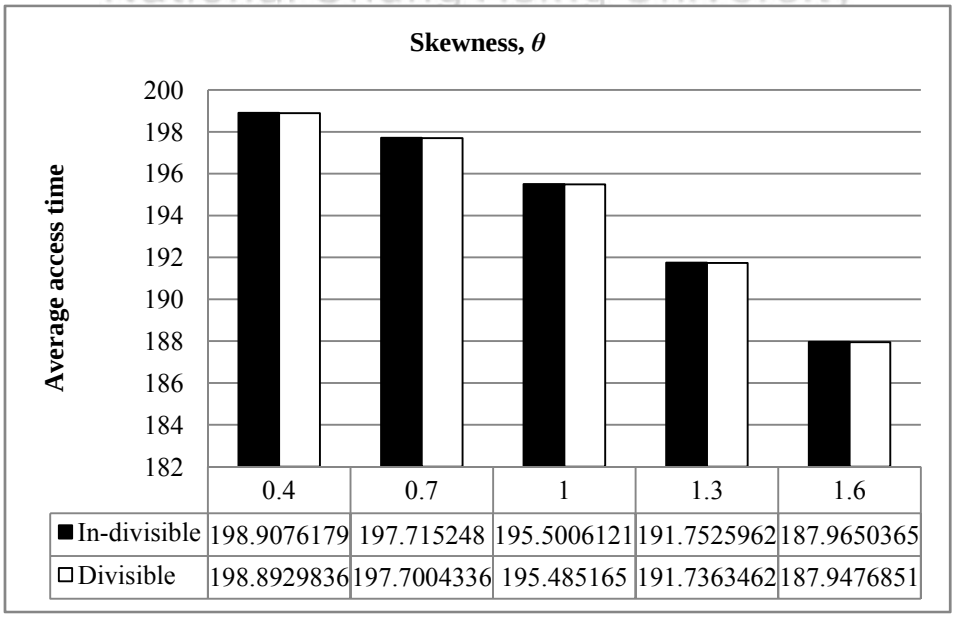
Constant parameters: $\lambda = 128$; $n = 1024$; $\theta = 1.0$; $N = 15$; $\gamma = 3.3$



	4	8	16	32	64
■ In-divisible	351.1935365	175.6425917	87.83603291	43.90030349	21.94401267
□ Divisible	351.1541701	175.6227018	87.82599155	43.895353	21.94152248

Figure 5.6. Effect of the number of channels with Poisson size distribution.

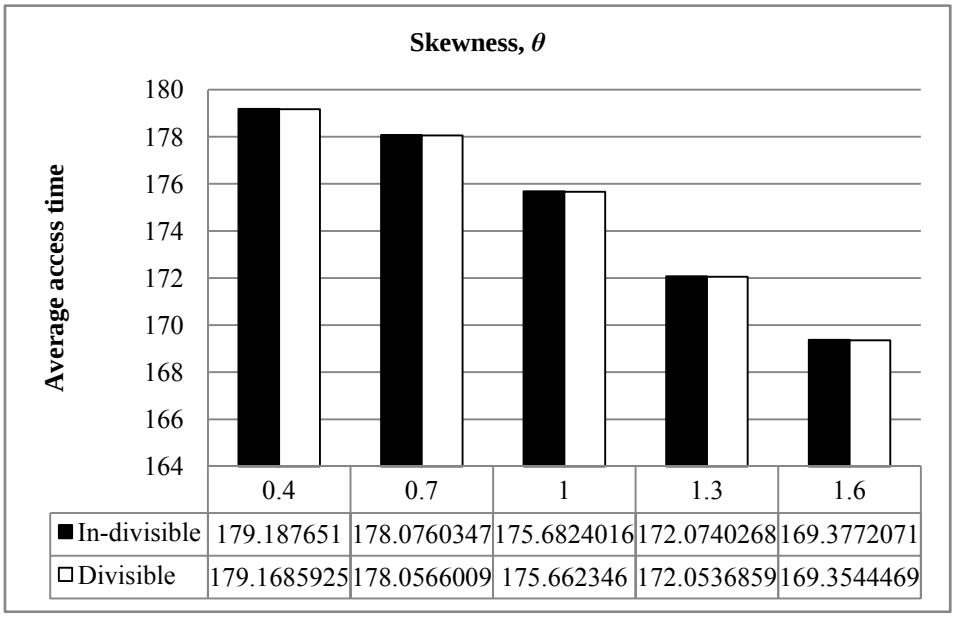
Constant parameters: $\lambda = 128$; $n = 1024$; $K = 8$; $N = 15$; $\gamma = 3.3$



	0.4	0.7	1	1.3	1.6
■ In-divisible	198.9076179	197.715248	195.5006121	191.7525962	187.9650365
□ Divisible	198.8929836	197.7004336	195.485165	191.7363462	187.9476851

Figure 5.7. Effect of the skewness with Normal size distribution.

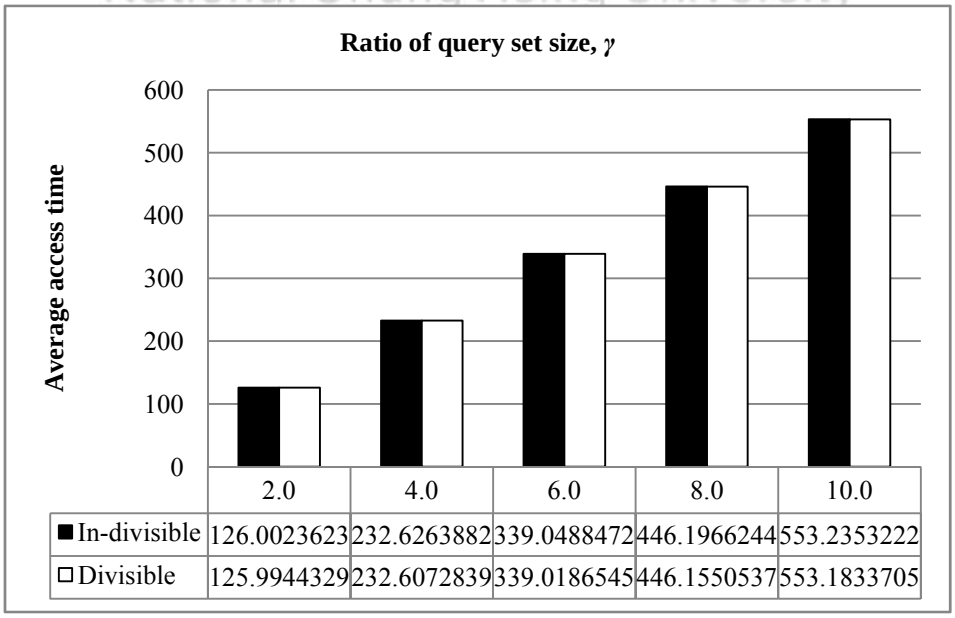
Constant parameters: $\lambda = 128$; $n = 1024$; $K = 8$; $N = 15$; $\gamma = 3.3$



	0.4	0.7	1	1.3	1.6
■ In-divisible	179.187651	178.0760347	175.6824016	172.0740268	169.3772071
□ Divisible	179.1685925	178.0566009	175.662346	172.0536859	169.3544469

Figure 5.8. Effect of the skewness with Poisson size distribution.

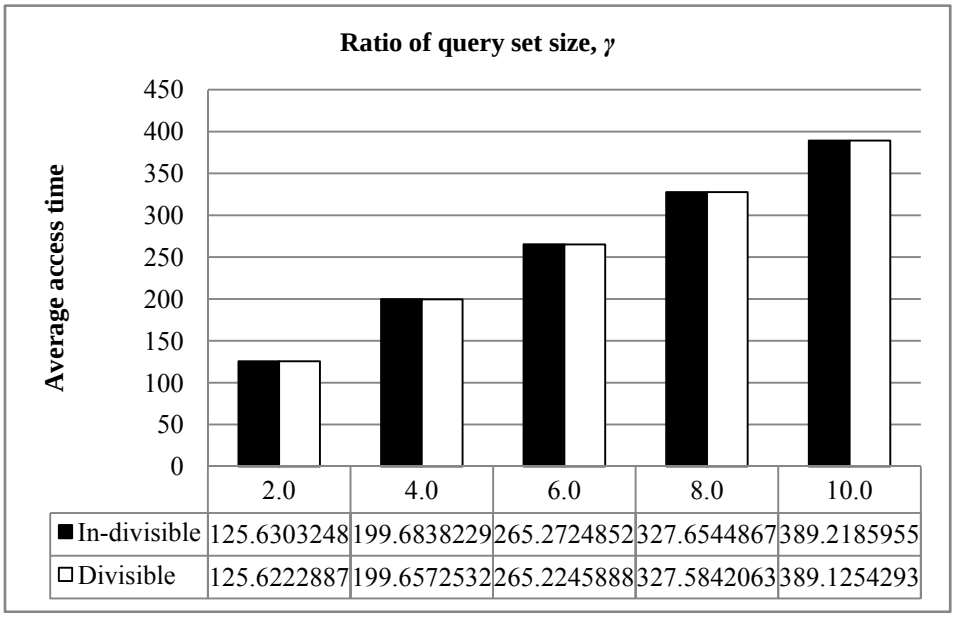
Constant parameters: $\lambda = 128$; $n = 1024$; $K = 8$; $\theta = 1.0$; $N = 15$



	2.0	4.0	6.0	8.0	10.0
■ In-divisible	126.0023623	232.6263882	339.0488472	446.1966244	553.2353222
□ Divisible	125.9944329	232.6072839	339.0186545	446.1550537	553.1833705

Figure 5.9. Effect of the ratio of query set size with Normal size distribution.

Constant parameters: $\lambda = 128$; $n = 1024$; $K = 8$; $\theta = 1.0$; $N = 15$



	2.0	4.0	6.0	8.0	10.0
■ In-divisible	125.6303248	199.6838229	265.2724852	327.6544867	389.2185955
□ Divisible	125.6222887	199.6572532	265.2245888	327.5842063	389.1254293

Figure 5.10. Effect of the ratio of query set size with Poisson size distribution.

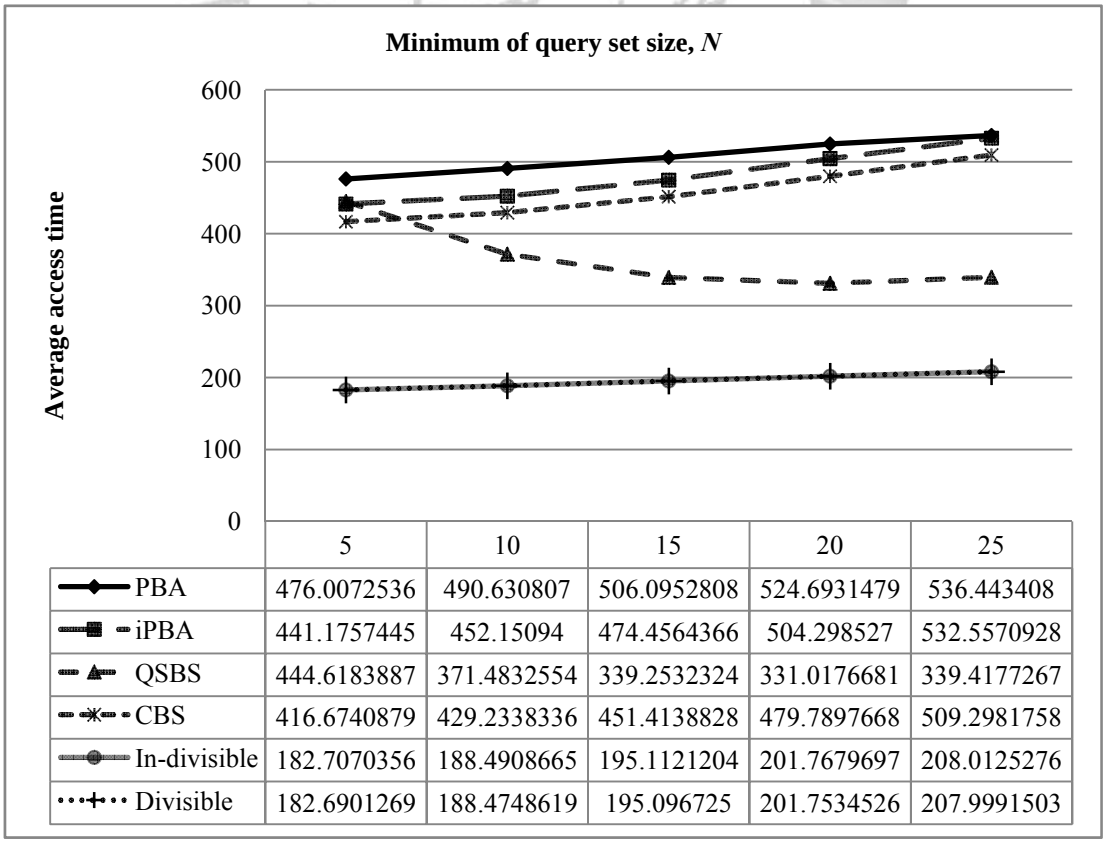
5.2. Effect of the minimum of query set size

In this section, two simulations show the effect of the minimal query set size with Normal and Poisson size distributions on the average access time for algorithms PBA, iPBA, CBS, QSBS. In Figures 5.11 and 5.12, the *y*-axis denotes the average access time, and the *x*-axis represents the value of *N*. The minimum of query set size *N* is limited from 5 to 25, and other parameters are set to the default values as shown in Table 5.1. Figures 5.11 and 5.12 show that the average access times of algorithms PBA, iPBA and CBS increase as *N* increases. In contrast, the average access time of algorithm QSBS decreases as *N* increases from 5 to 15 and its curve nearly parallels that of the lower bounds as *N* is greater than 15. The performance of QSBS is better than that of others.

Since the size of each query set is greater than or equal to *N*, it has to take more access time to retrieve each query set as *N* increases. As a result, the average access times of algorithms PBA, iPBA and CBS increase as *N* increases. Among them, iPBA

which keeps a better scheduling priority, slightly outperforms PBA, while CBS which keeps a better coherence degree, slightly outperforms iPBA. QSBS keeps both the scheduling priority and the best coherence degree by data-item replication; therefore, it outperforms others on the average access time. However, QSBS performs worse than others when N equals 5 as shown in Figures 5.11 and 5.12. According to our observation, algorithm QSBS schedules most of query sets into the last two channels as the size differential is very large, and this anomaly causes a worse average access time. The smaller N makes the larger size differential; therefore, QSBS performs worse than others as N is very small.

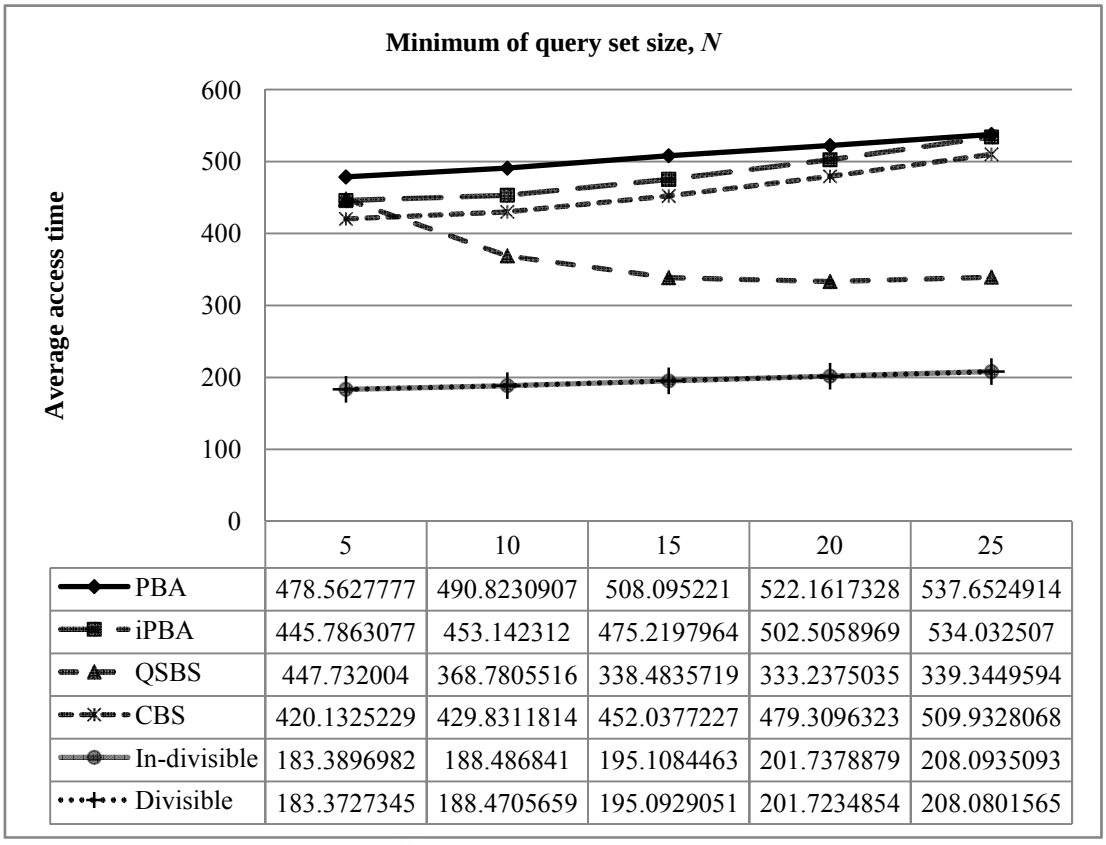
Constant parameters: $\lambda = 128$; $n = 1024$; $K = 8$; $\theta = 1.0$; $\gamma = 3.3$



	5	10	15	20	25
PBA	476.0072536	490.630807	506.0952808	524.6931479	536.443408
iPBA	441.1757445	452.15094	474.4564366	504.298527	532.5570928
QSBS	444.6183887	371.4832554	339.2532324	331.0176681	339.4177267
CBS	416.6740879	429.2338336	451.4138828	479.7897668	509.2981758
In-divisible	182.7070356	188.4908665	195.1121204	201.7679697	208.0125276
Divisible	182.6901269	188.4748619	195.096725	201.7534526	207.9991503

Figure 5.11. Effect of the minimal query set size with Normal size distribution.

Constant parameters: $\lambda = 128$; $n = 1024$; $K = 8$; $\theta = 1.0$; $\gamma = 3.3$



	5	10	15	20	25
PBA	478.5627777	490.8230907	508.095221	522.1617328	537.6524914
iPBA	445.7863077	453.142312	475.2197964	502.5058969	534.032507
QSBS	447.732004	368.7805516	338.4835719	333.2375035	339.3449594
CBS	420.1325229	429.8311814	452.0377227	479.3096323	509.9328068
In-divisible	183.3896982	188.486841	195.1084463	201.7378879	208.0935093
Divisible	183.3727345	188.4705659	195.0929051	201.7234854	208.0801565

Figure 5.12. Effect of the minimal query set size with Poisson size distribution.

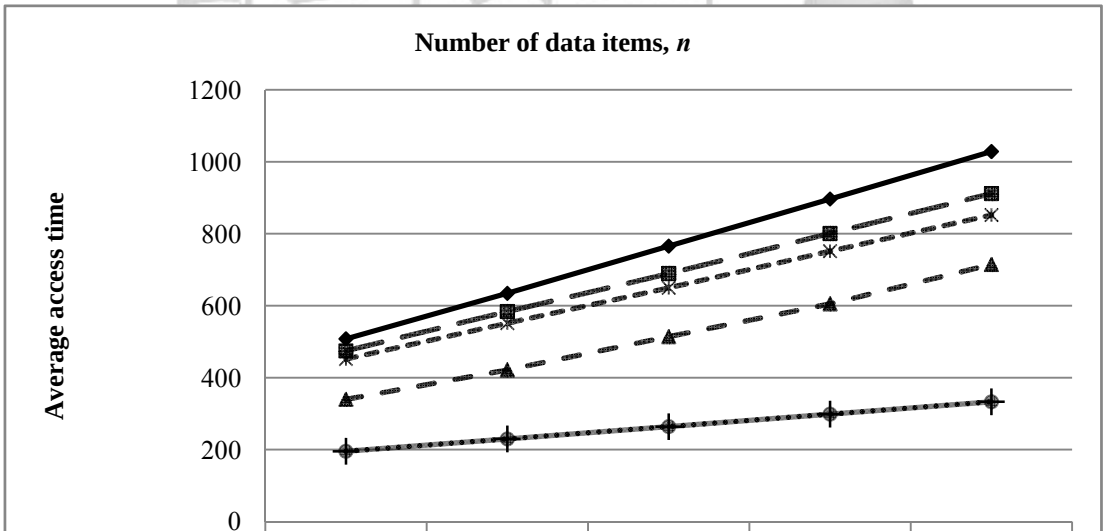
5.3. Effect of the number of data items

In this section, two simulations show the effect of the number of data items with Normal and Poisson size distribution on the average access time for algorithms PBA, iPBA, CBS, QSBS. In Figures 5.13 and 5.14, the *y*-axis denotes the average access time, and the *x*-axis represents the value of *n*. The number of data items *n* is limited from 1024 to 1824, and other parameters are set to the default values as shown in Table 5.1. Figures 5.13 and 5.14 show that the average access times of all algorithms increase as *n* increases. In Figure 5.13, the performance of iPBA is only better than that of PBA, CBS slightly outperforms iPBA, and QSBS shows the best performance among all algorithms. However, QSBS is getting worse than iPBA and CBS as *n* is greater than

1424 in Figure 5.14.

Since the increase of the data item number causes the longer cycle length on each broadcast channel, it will take more average access time to retrieve a data item in each query set. That is the reason why the curve of each algorithm ascends as n increases in Figures 5.13 and 5.14. The reason of the poor performance of QSBS is that the Poisson size distribution is a skewed distribution and will cause a greater size differential among all query sets. For instance, {0.3, 0.7} is a skewed size distribution of two query sets, and the number of data items n are 100 and 500, respectively. The query set sizes are 30 and 70 as n equals 100. The other two sizes are 150 and 350 as n equals 500. Therefore, the size differentials are 40 and 200 under the two numbers of data items.

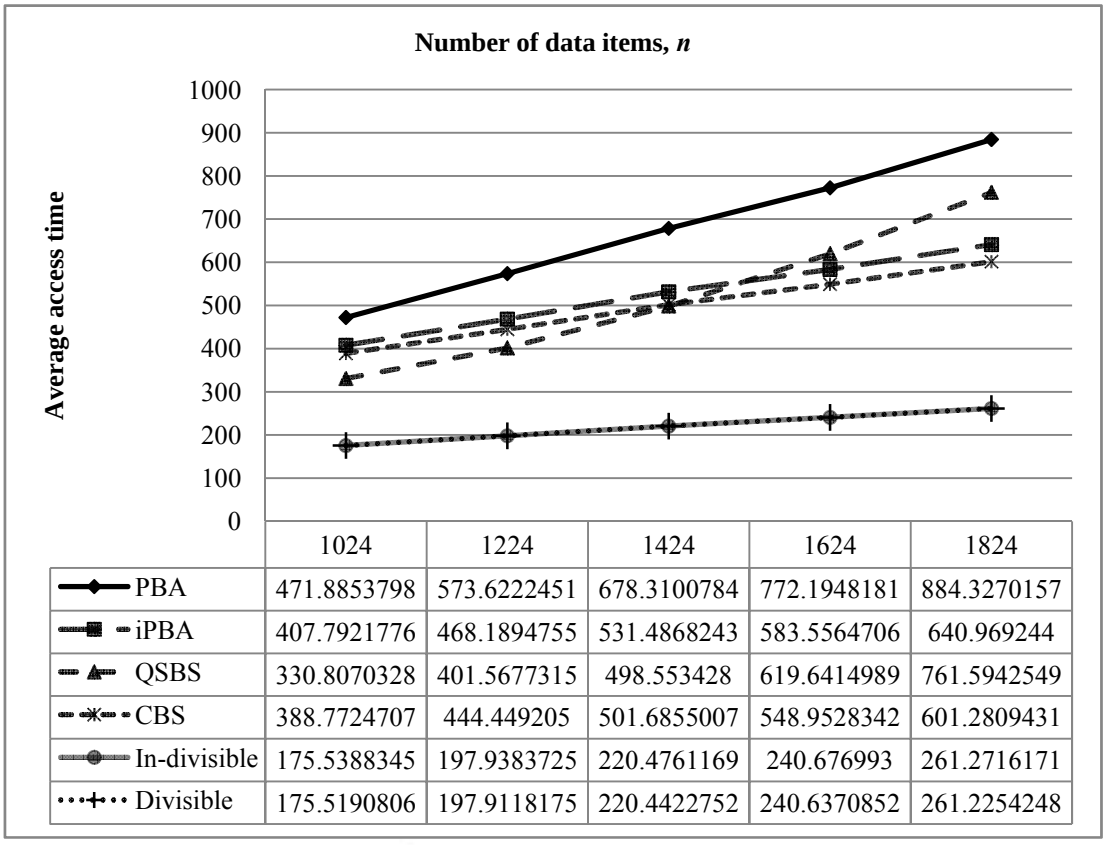
Constant parameters: $\lambda = 128$; $K = 8$; $\theta = 1.0$; $N = 15$; $\gamma = 3.3$



	1024	1224	1424	1624	1824
PBA	507.7240716	634.2441977	765.4924921	896.3551194	1028.354762
iPBA	474.2397308	583.7767749	689.3417795	800.7045346	911.635454
QSBS	339.3768176	421.4563691	513.7006823	605.1664664	714.5330281
CBS	451.6406205	551.2105127	649.5113183	751.6497127	852.0288082
In-divisible	195.2551468	229.6255077	263.6627946	298.4366763	332.8357173
Divisible	195.239853	229.6064977	263.6402103	298.4103205	332.8057539

Figure 5.13. Effect of the number of data items with Normal size distribution.

Constant parameters: $\lambda = 128$; $K = 8$; $\theta = 1.0$; $N = 15$; $\gamma = 3.3$



	1024	1224	1424	1624	1824
PBA	471.8853798	573.6222451	678.3100784	772.1948181	884.3270157
iPBA	407.7921776	468.1894755	531.4868243	583.5564706	640.969244
QSBS	330.8070328	401.5677315	498.553428	619.6414989	761.5942549
CBS	388.7724707	444.449205	501.6855007	548.9528342	601.2809431
In-divisible	175.5388345	197.9383725	220.4761169	240.676993	261.2716171
Divisible	175.5190806	197.9118175	220.4422752	240.6370852	261.2254248

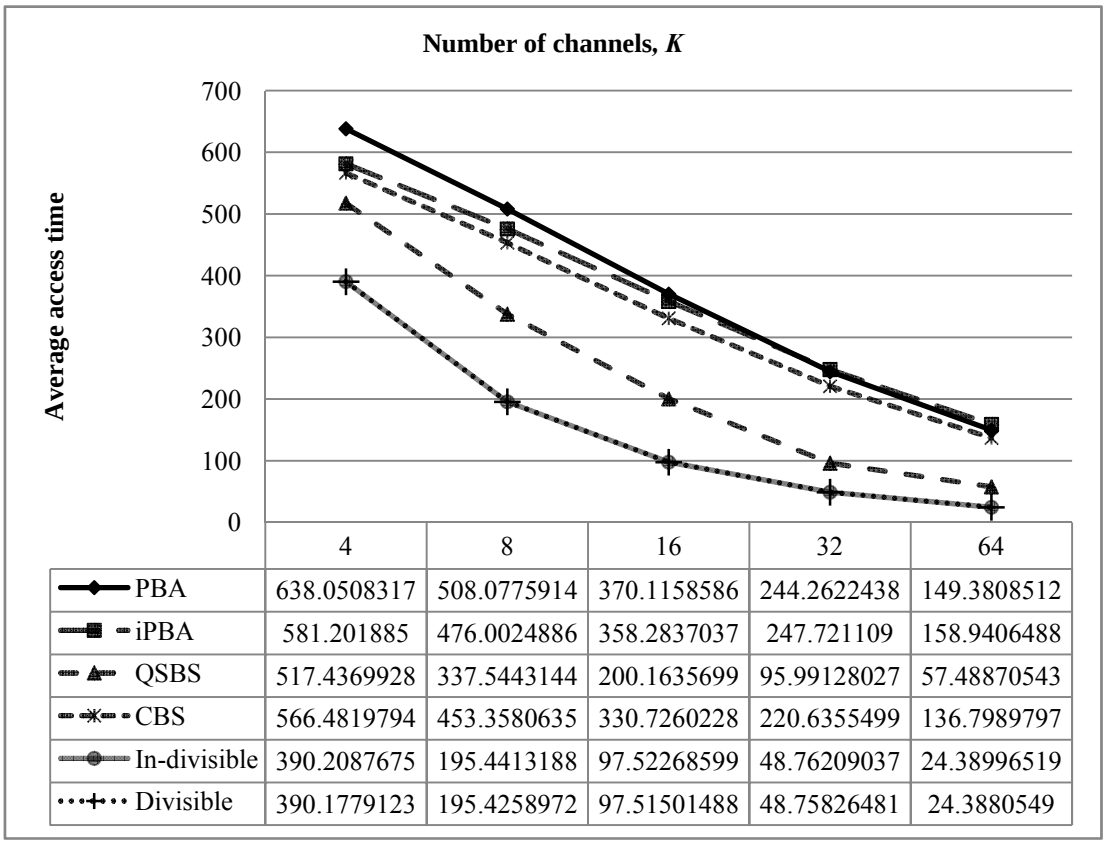
Figure 5.14. Effect of the number of data items with Poisson size distribution.

5.4. Effect of the number of broadcast channels

In this section, two simulations show the effect of the number of channels with Normal and Poisson size distribution on the average access time for algorithms PBA, iPBA, CBS, QSBS. In Figures 5.15 and 5.16, the *y*-axis denotes the average access time, and the *x*-axis represents the value of *K*. The number of channels *K* is limited from 4 to 64, and other parameters are set to the default values as shown in Table 5.1. Figures 5.15 and 5.16 show that, the average access times of all algorithms decrease as *K* increases. QSBS outperforms the others and is close to the lower bound as *K* is large. More broadcast channels will cause shorter cycle length on each channel. Therefore, it takes less average access time to retrieve each data item in query sets from a channel

with shorter cycle length.

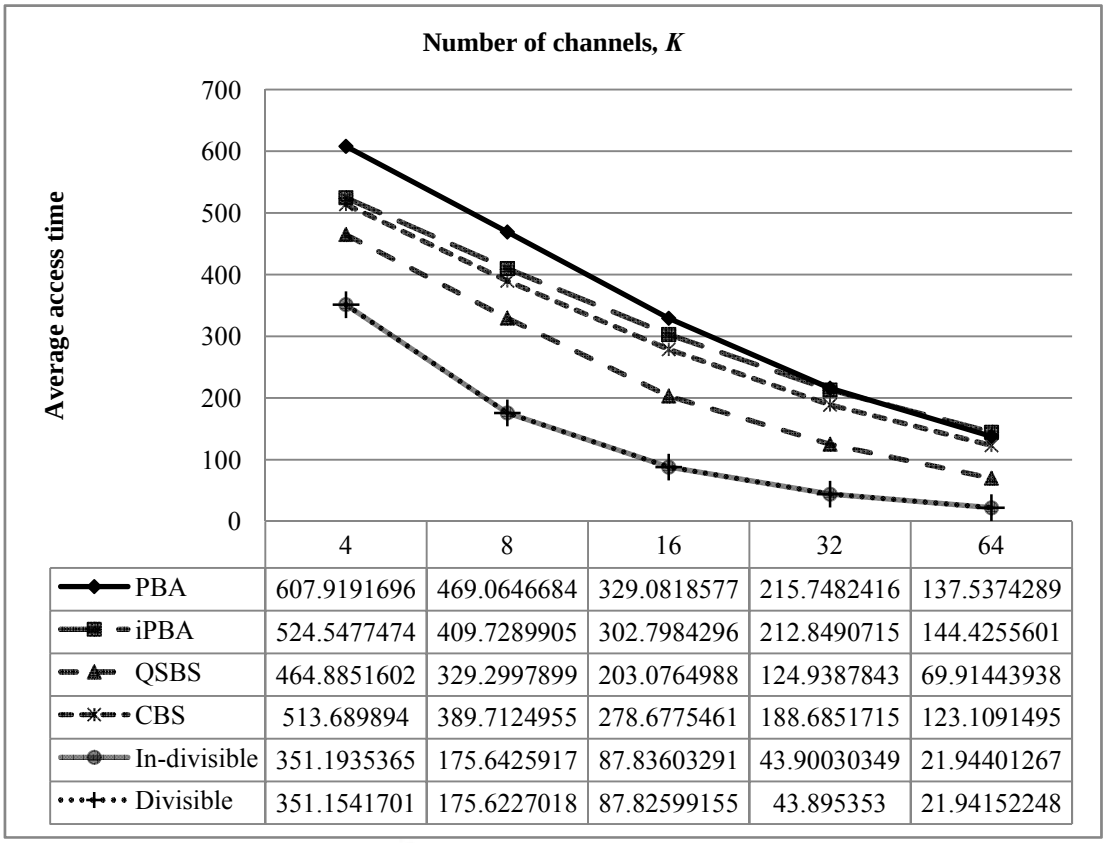
Constant parameters: $\lambda = 128$; $n = 1024$; $\theta = 1.0$; $N = 15$; $\gamma = 3.3$



	4	8	16	32	64
PBA	638.0508317	508.0775914	370.1158586	244.2622438	149.3808512
iPBA	581.201885	476.0024886	358.2837037	247.721109	158.9406488
QSBS	517.4369928	337.5443144	200.1635699	95.99128027	57.48870543
CBS	566.4819794	453.3580635	330.7260228	220.6355499	136.7989797
In-divisible	390.2087675	195.4413188	97.52268599	48.76209037	24.38996519
Divisible	390.1779123	195.4258972	97.51501488	48.75826481	24.3880549

Figure 5.15. Effect of the number of channels with Normal size distribution.

Constant parameters: $\lambda = 128$; $n = 1024$; $\theta = 1.0$; $N = 15$; $\gamma = 3.3$



	4	8	16	32	64
PBA	607.9191696	469.0646684	329.0818577	215.7482416	137.5374289
iPBA	524.5477474	409.7289905	302.7984296	212.8490715	144.4255601
QSBS	464.8851602	329.2997899	203.0764988	124.9387843	69.91443938
CBS	513.689894	389.7124955	278.6775461	188.6851715	123.1091495
In-divisible	351.1935365	175.6425917	87.83603291	43.90030349	21.94401267
Divisible	351.1541701	175.6227018	87.82599155	43.895353	21.94152248

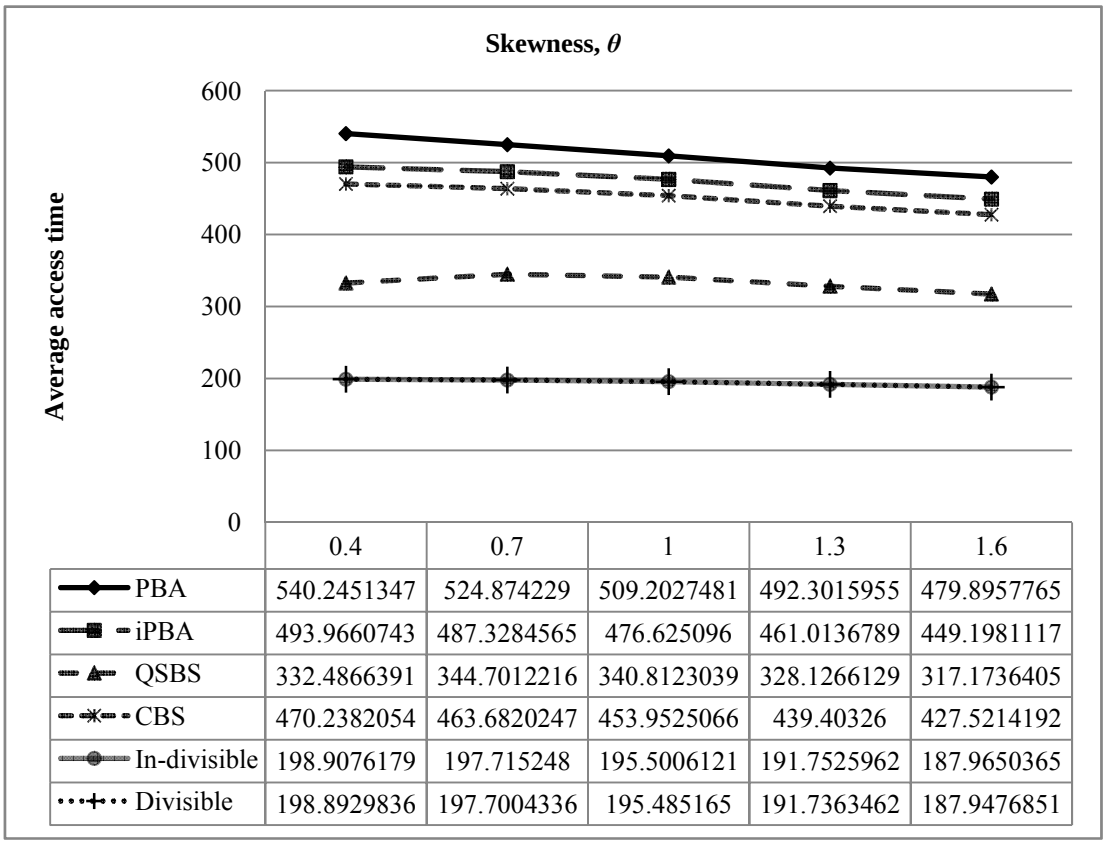
Figure 5.16. Effect of the number of channels with Poisson size distribution.

5.5. Effect of the skewness

In this section, two simulations show the effect of the skewness with Normal and Poisson size distribution on the average access time for algorithms PBA, iPBA, CBS, QSBS. In Figures 5.17 and 5.18, the *y*-axis represents the average access time, and the *x*-axis denotes the value of skewness. The value of θ is limited from 0.4 to 1.6, and other parameters are set to the default values as shown in Table 5.1. Figures 5.17 and 5.18 illustrate that the average access time of all algorithms decrease as the value of θ increases. When θ increases, the access probabilities are increasingly skewed. A high skew access probabilities means that a small number of query sets are accessed frequently. This explains why the average access time decreases as shown in Figures

5.17 and 5.18.

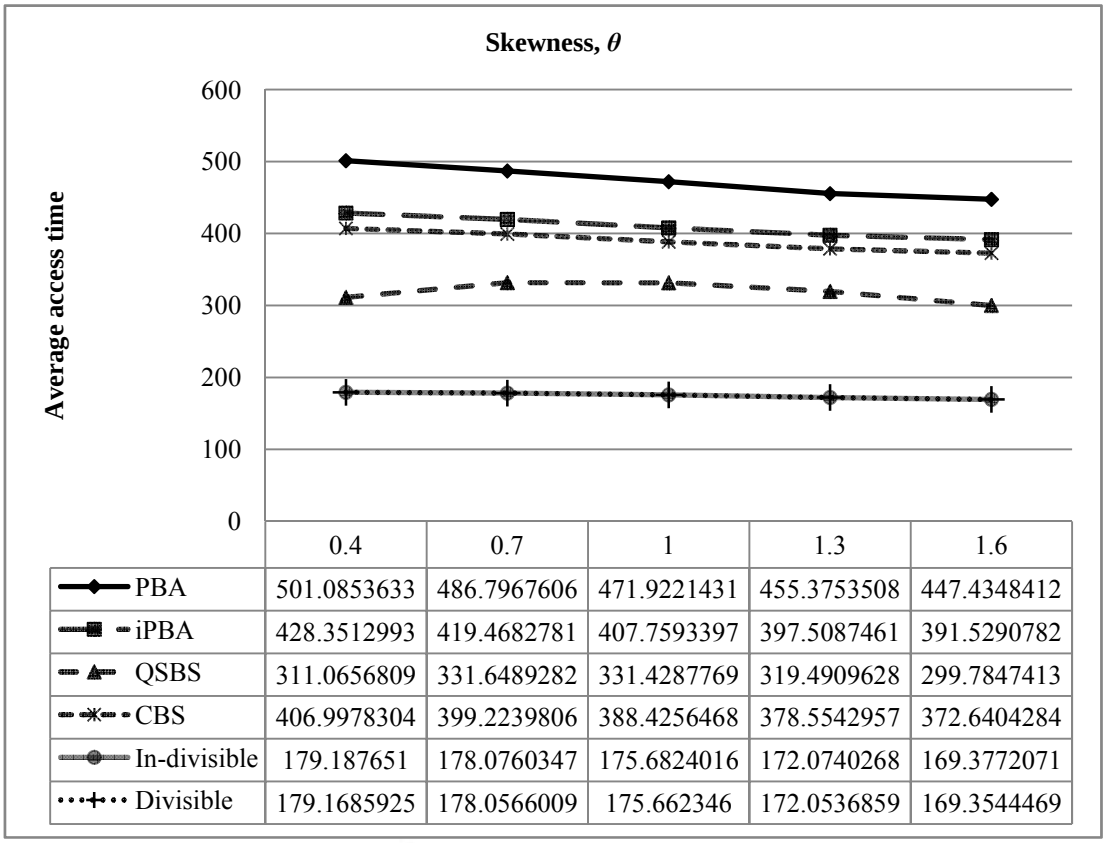
Constant parameters: $\lambda = 128$; $n = 1024$; $K = 8$; $N = 15$; $\gamma = 3.3$



	0.4	0.7	1	1.3	1.6
PBA	540.2451347	524.874229	509.2027481	492.3015955	479.8957765
iPBA	493.9660743	487.3284565	476.625096	461.0136789	449.1981117
QSBS	332.4866391	344.7012216	340.8123039	328.1266129	317.1736405
CBS	470.2382054	463.6820247	453.9525066	439.40326	427.5214192
In-divisible	198.9076179	197.715248	195.5006121	191.7525962	187.9650365
Divisible	198.8929836	197.7004336	195.485165	191.7363462	187.9476851

Figure 5.17. Effect of the skewness with Normal size distribution.

Constant parameters: $\lambda = 128$; $n = 1024$; $K = 8$; $N = 15$; $\gamma = 3.3$



	0.4	0.7	1	1.3	1.6
PBA	501.0853633	486.7967606	471.9221431	455.3753508	447.4348412
iPBA	428.3512993	419.4682781	407.7593397	397.5087461	391.5290782
QSBS	311.0656809	331.6489282	331.4287769	319.4909628	299.7847413
CBS	406.9978304	399.2239806	388.4256468	378.5542957	372.6404284
In-divisible	179.187651	178.0760347	175.6824016	172.0740268	169.3772071
Divisible	179.1685925	178.0566009	175.662346	172.0536859	169.3544469

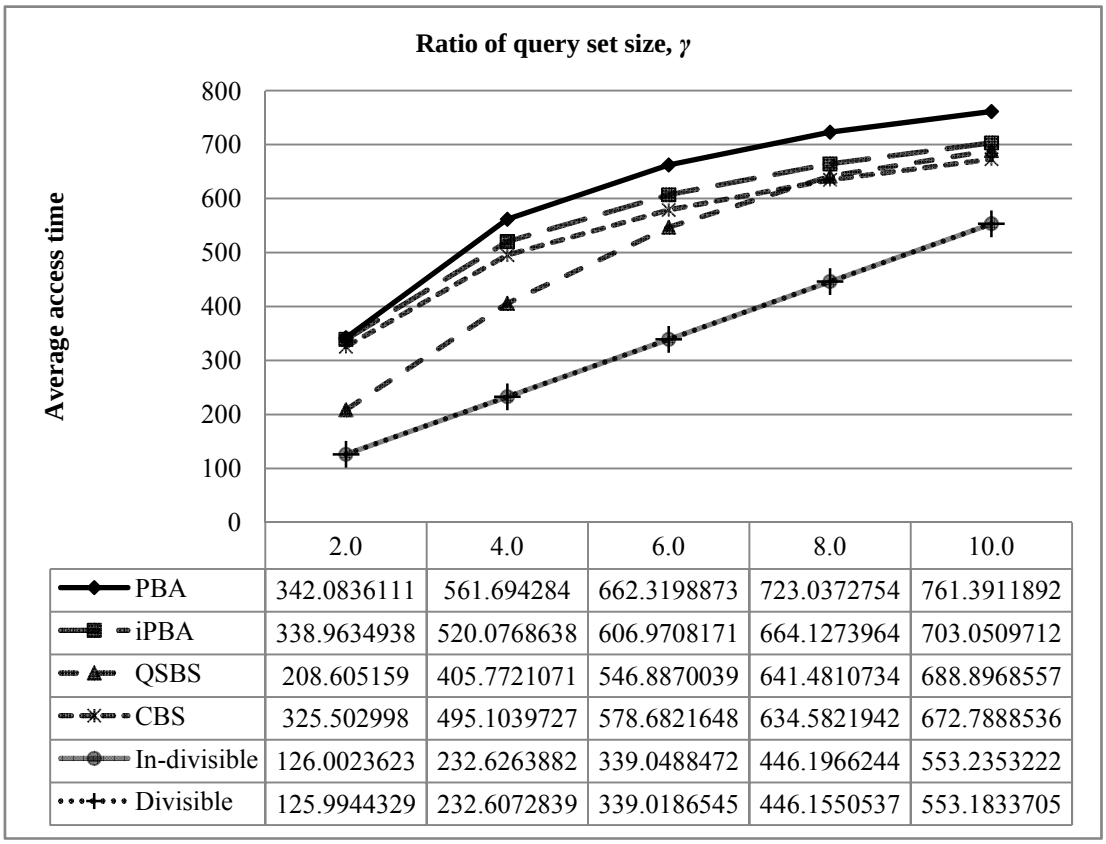
Figure 5.18. Effect of the skewness with Poisson size distribution.

5.6. Effect of the ratio of query set size

In this section, two simulations show the effect of the ratio of query set size to data item number with Normal and Poisson size distribution on the average access time for algorithms PBA, iPBA, CBS, QSBS. In Figures 5.19 and 5.20, the *y*-axis represents the average access time, and the *x*-axis denotes the value of γ. The value of γ is limited from 2.0 to 10.0, and other parameters are set to the default values as shown in Table 5.1. Figure 5.19 and Figure 5.20 illustrate that the curve of each algorithm ascends as γ increases. In Figure 5.19, QSBS is worse than CBS as γ is greater than 8.0. QSBS is worse than the others as γ is greater than 7.0. The reason is that a higher value of γ causes a higher degree of size differential among all query sets, and the influence is

more obvious especially under Poisson distribution.

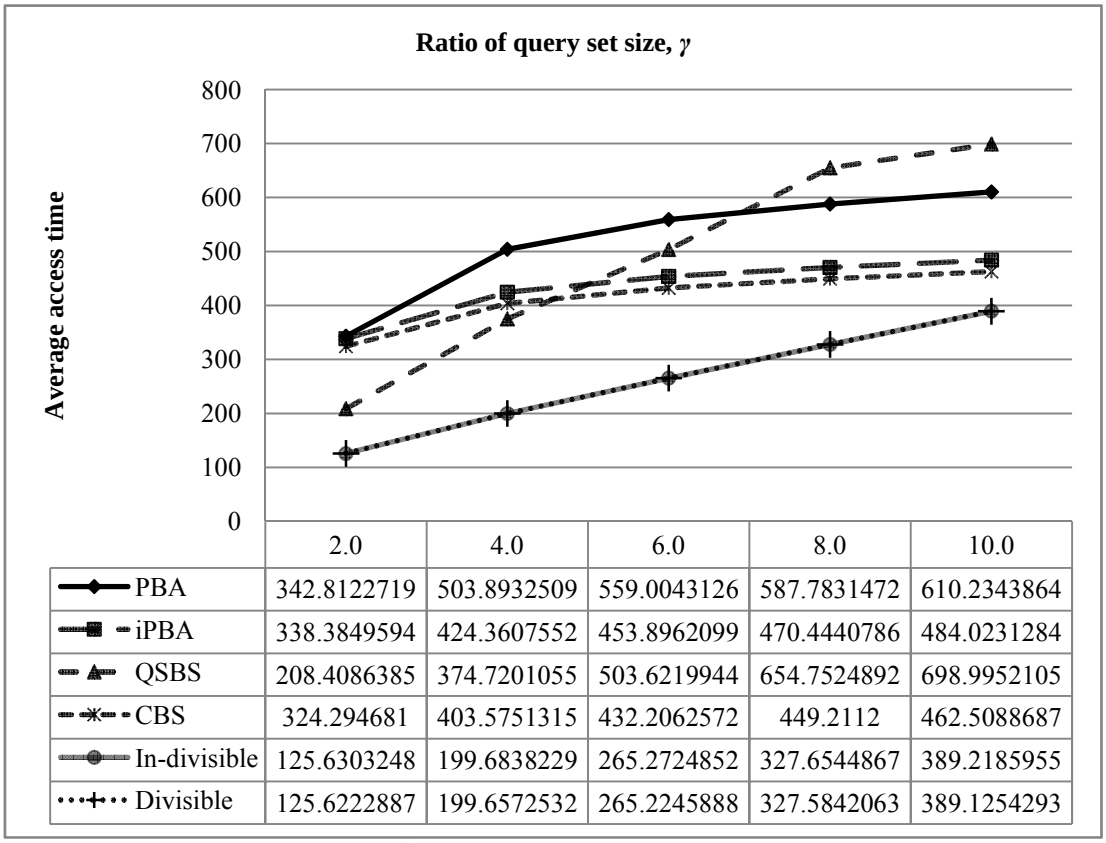
Constant parameters: $\lambda = 128$; $n = 1024$; $K = 8$; $\theta = 1.0$; $N = 15$



	2.0	4.0	6.0	8.0	10.0
PBA	342.0836111	561.694284	662.3198873	723.0372754	761.3911892
iPBA	338.9634938	520.0768638	606.9708171	664.1273964	703.0509712
QSBS	208.605159	405.7721071	546.8870039	641.4810734	688.8968557
CBS	325.502998	495.1039727	578.6821648	634.5821942	672.7888536
In-divisible	126.0023623	232.6263882	339.0488472	446.1966244	553.2353222
Divisible	125.9944329	232.6072839	339.0186545	446.1550537	553.1833705

Figure 5.19. Effect of the ratio of query set size with Normal size distribution.

Constant parameters: $\lambda = 128$; $n = 1024$; $K = 8$; $\theta = 1.0$; $N = 15$



	2.0	4.0	6.0	8.0	10.0
PBA	342.8122719	503.8932509	559.0043126	587.7831472	610.2343864
iPBA	338.3849594	424.3607552	453.8962099	470.4440786	484.0231284
QSBS	208.4086385	374.7201055	503.6219944	654.7524892	698.9952105
CBS	324.294681	403.5751315	432.2062572	449.2112	462.5088687
In-divisible	125.6303248	199.6838229	265.2724852	327.6544867	389.2185955
Divisible	125.6222887	199.6572532	265.2245888	327.5842063	389.1254293

Figure 5.20. Effect of the ratio of query set size with Poisson size distribution.

It is observed from the experimental results that, each query set in a query profile is partitioned into at most two adjacent channels by algorithms CBS and QSBS. In Lemma 4, the condition of partitioning a query set into more than two groups has been derived. Inequality (4.25) shows three significant factors, λ, K and n_j, affect the times of partition. In all simulations, λ is fixed, and K and n_j are variable. Hence, Inequality (4.25) can be met with a large K and a large n_j. However, K, is always far smaller than λ, and the algorithms CBS and QSBS will schedule all data items in the query set with a large n_j into the last two channels. As a result, each query set cannot be partitioned into more than two groups.

6 Conclusions and future work

Data broadcast is an efficient approach to serve data access among an unlimited number of mobile users. In this thesis, an investigation of broadcast-program scheduling algorithms is performed for querying multiple data items in the multiple channels environment. The goal of this study is to reduce the average access time of all mobile users. Three algorithms including iPBA , CBS and QSBS are proposed. Algorithm iPBA extends algorithm PBA to considering the query set size. Algorithm CBS, a coherence-based algorithm, generates a broadcast program on multiple channels without data-item replication. Algorithm QSBS treats data items in each query set as an aggregate data item (CDI) and performs data scheduling with query-set replication.

In the experiments, algorithm iPBA slightly outperforms algorithm PBA, i.e., the importance of query set size is justified. Algorithm CBS also slightly outperforms algorithms PBA and iPBA in all experiments; hence, the property of coherence plays an important role. Algorithm QSBS outperforms the other algorithms a lot when the size differential among all query sets is not too large.

In this study, we use the query-set-based concept to develop algorithm QSBS. Mobile users can retrieve all data items in a query set from at most two broadcast channels. That is, they do not need to switch channels frequently in each request. Furthermore, the simulations confirm that the average access time is reduced by QSBS.

The future research will focus on eliminating the effect of size differential from algorithm QSBS and minimizing the average access time. Algorithm QSBS outperforms others very much, but it is affected by the size differential. The reason is that the original formula of cycle length will cause the extreme length of broadcast cycle in the last two channels. Therefore, we have to take the size distribution into consideration and

design a new approach to deal with skewed size distributions. In addition, algorithm CBS stably possesses a little advantage on the average access time over the others, but it is not affected by the size differential. A hybrid algorithm integrating the concepts of both algorithms CBS and QSBS may be appropriate for all kinds of size distributions. To develop such a hybrid algorithm, a fast parsing mechanism has to be designed to detect the size differential over all query sets. As long as a very skewed size distribution is detected, the server schedules data with algorithm CBS instead.

References

[1] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik, "Broadcast Disks: Data Management for Asymmetric Communication Environments," *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Cata*, pp. 199-210 1995.

[2] S. Acharya, M. Franklin, and S. Zdonik, "Balancing Push and Pull for Data Broadcast," *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, pp. 183-194, 1997.

[3] S. Acharya and S. Muthukrishnan, "Scheduling On-Demand Broadcasts: New Metrics and Algorithms," *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 43-54, 1998.

[4] D. Aksoy and M. Franklin, "Scheduling for Large-Scale On-Demand Data Broadcasting," *Proceedings IEEE INFOCOM '98 Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies*, pp. 651-659, 1998.

[5] D. Aksoy, M. J. Franklin, and S. B. Zdonik, "Data Staging for On-Demand Broadcast," *Proceedings of the 27th International Conference on Very Large Data Bases*, pp. 571-580, 2001.

[6] AoPSWiki, "http://www.artofproblemsolving.com/Wiki/index.php/Multinomial_Theorem," 2008.

[7] E. Ardizzoni, A. A. Bertossi, M. C. Pinotti, S. Ramaprasad, R. Rizzi, and M. V. S. Shashanka, "Optimal Skewed Data Allocation on Multiple Channels with Flat

Broadcast per Channel," *IEEE Transactions on Computers,* vol. 54, pp. 558-572, 2005.

[8] A. Bar-Noy, J. S. Naor, and B. Schieber, "Pushing Dependent Data in Clients-Providers-Servers Systems," *Wireless Networks,* vol. 9, pp. 421-430, 2003.

[9] A. Bar-Noy and Y. Shilo, "Optimal Broadcasting of Two Files over an Asymmetric Channel," *Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies*, pp. 267 - 274, 1999.

[10] J.-C. Bolot, "End-to-End Packet Delay and Loss Behavior in the Internet," *Proceedings on Communications Architectures, Protocols and Applications*, pp. 289-298, 1993.

[11] Y.-I. Chang and W.-H. Hsieh, "An Efficient Scheduling Method for Query-Set-Based Broadcasting in Mobile Environments," *Proceedings of the 24th International Conference on Distributed Computing Systems Workshops*, pp. 478-483, 2004.

[12] Y. D. Chung and M. H. Kim, "QEM: A Scheduling Method for Wireless Broadcast Data," *Proceedings of the 6th International Conference on Database Systems for Advanced Applications*, pp. 135-142, 1999.

[13] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley, 1989.

[14] S. Hameed and N. H. Vaidya, "Efficient Algorithms for Scheduling Data Broadcast," *Wireless Networks,* vol. 5, pp. 183-193, 1999.

[15] S. Hameed and N. H. Vaidya, "Log-time Agorithms for Sheduling Single and Multiple Channel Data Broadcast," *Proceedings of the 3rd Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 90-99,

1997.

[16] C.-H. Hsu, G. Lee, and A. L. Chen, "An Efficient Algorithm for Near Optimal Data Allocation on Multiple Broadcast Channels," *Distributed and Parallel Databases,* vol. 18, pp. 207-222, 2005.

[17] C.-L. Hu and M.-S. Chen, "Adaptive Multichannel Data Dissemination: Support of Dynamic Traffic Awareness and Push-Pull Time Balance," *IEEE Transactions on Vehicular Technology,* vol. 54, pp. 673-686, 2005.

[18] J.-L. Huang and M.-S. Chen, "Broadcast Program Generation for Unordered Queries with Data Replication," *Proceedings of the 2003 ACM Symposium on Applied Computing*, pp. 866-870, 2003.

[19] J.-L. Huang and M.-S. Chen, "Dependent Data Broadcasting for Unordered Queries in a Multiple Channel Mobile Environment," *IEEE Transactions on Knowledge and Data Engineering,* vol. 16, pp. 1143-1156, 2004.

[20] J.-L. Huang, M.-S. Chen, and W.-C. Peng, "Broadcasting Dependent Data for Ordered Queries without Replication in a Multi-Channel Mobile Environment," *Proceedings of the 19th International Conference on Data Engineering*, pp. 692-694, 2003.

[21] J.-L. Huang and W.-C. Peng, "An Effective Broadcast Program Generation Algorithm for Dependent Data," *Proceedings of the Emerging Information Technology Conference*, 4 pp., 2005.

[22] J.-L. Huang, W.-C. Peng, and M.-S. Chen, "SOM: Dynamic Push-pull Channel Allocation Framework for Mobile Data Broadcasting," *IEEE Transactions on Mobile Computing,* vol. 5, pp. 974-990, 2006.

[23] H.-P. Hung and M.-S. Chen, "MULS: A General Framework of Providing Multilevel Service Quality in Sequential Data Broadcasting," *IEEE Transactions*

on Knowledge and Data Engineering, vol. 19, pp. 1433-1447, 2007.

[24] H.-P. Hung and M.-S. Chen, "On Exploring Channel Allocation in the Diverse Data Broadcasting Environment," *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems*, pp. 729-738, 2005.

[25] H.-P. Hung, J.-W. Huang, J.-L. Huang, and M.-S. Chen, "Scheduling Dependent Items in Data Broadcasting Environments," *Proceedings of the 2006 ACM Symposium on Applied Computing*, pp. 1177-1181, 2006.

[26] J.-J. Hung and A. Seifert, "FlexSched: A Parameterized Data Schedule Generator for Multi-Channel Broadcast Systems," *Proceedings of the 7th International Conference on Mobile Data Management*, pp. 129-129, 2006.

[27] R. E. Johnson, F. L. Kiokemeister, and E. S. Wolk, *Calculus with Analytic Geometry*, Sixth ed, 1978.

[28] S. H. Kang, S. Choi, S. J. Choi, G. Lee, J. Lew, and J. Lee, "Scheduling Data Broadcast Based on Multi-Frequency in Mobile Interactive Broadcasting," *IEEE Transactions on Broadcasting,* vol. 53, pp. 405-411, 2007.

[29] G. Lee and S.-C. Lo, "Broadcast Data Allocation for Efficient Access of Multiple Data Items in Mobile Environments," *Mobile Networks and Applications,* vol. 8, pp. 365-375, 2003.

[30] K.-F. Lin and C.-M. Liu, "Broadcasting Dependent Data with Minimized Access Latency in a Multi-Channel Environment," *Proceedings of the 2006 International Conference on Wireless Communications and Mobile Computing*, pp. 809-814, 2006.

[31] F. J. Ovalle-Martínez, J. S. González, and I. Stojmenović, "A Parallel Hill Climbing Algorithm for Pushing Dependent Data in Clients-Providers-Servers Systems," *Mobile Networks and Applications,* vol. 9, pp. 257-264, 2004.

[32] W.-C. Peng and M.-S. Chen, "Dynamic Generation of Data Broadcasting Programs for a Broadcast Disk Array in a Mobile Computing Environment," *Proceedings of the 9th International Conference on Information and knowledge Management*, pp. 38-45, 2000.

[33] C.-J. Su, L. Tassiulas, and V. J. Tsotras, "Broadcast Scheduling for Information Distribution," *Wireless Networks,* vol. 5, pp. 137-147, 1999.

[34] E. Tiakas, S. Ougiaroglou, and P. Nicopolitidis, "Efficient Broadcast Disks Program Construction in Asymmetric Communication Environments," *Proceedings of the 10th ACM Symposium on Modeling, Analysis, and Simulation of Wireless and Mobile Systems*, pp. 279-283, 2007.

[35] N. H. Vaidya and S. Hameed, "Scheduling Data Broadcast in Asymmetric Communication Environments," *Wireless Networks,* vol. 5, pp. 171-182, 1999.

[36] S. Wang and H.-L. Chen, "Near-optimal Data Allocation over Multiple Broadcast Channels," *Computer Communications,* vol. 29, pp. 1341-1349, 2006.

[37] J. W. Wong, "Broadcast Delivery," *Proceedings of the IEEE,* vol. 76, pp. 1566-1577, 1998.

[38] G.-M. Wu, "An Efficient Data Placement for Query-Set-Based Broadcasting in Mobile Environments," *Computer Communications,* vol. 30, pp. 1075-1081, 2007.

[39] W. G. Yee, S. B. Navathe, E. Omiecinski, and C. Jermaine, "Efficient Data Allocation over Multiple Channels at Broadcast Servers," *IEEE Transactions on Computers,* vol. 51, pp. 1231-1236, 2002.