MULS: A General Framework of Providing Multilevel Service Quality in Sequential Data Broadcasting

Hao-Ping Hung and Ming-Syan Chen, *Fellow*, *IEEE*

Abstract—In recent years, data broadcasting has become a promising technique to design a mobile information system with power conservation, high scalability, and high bandwidth utilization. In many applications, the query issued by a mobile client corresponds to multiple items that should be accessed in a *sequential* order. In this paper, we study the scheduling approach in such a *sequential data broadcasting* environment. Explicitly, we propose a general framework referred to as MULS (standing for MUltiLevel Service) for an information system. There are two primary stages in MULS: online scheduling (OLS) and optimization procedure. In the first stage, we propose an OLS algorithm to allocate the data items into multiple channels. As for the second stage, we devise an optimization procedure, called Sampling with Controlled Iteration (SCI), to enhance the quality of broadcast programs generated by algorithm OLS. Procedure SCI is able to strike a compromise between effectiveness and efficiency by tuning the control parameters. According to the experimental results, we show that algorithm OLS with procedure SCI outperforms the approaches in prior works prominently in both effectiveness (that is, the average access time of mobile users) and efficiency (that is, the complexity of the scheduling algorithm). Therefore, by cooperating algorithm OLS with procedure SCI, the proposed MULS framework is able to generate broadcast programs with the flexibility of providing different service qualities under different requirements of effectiveness and efficiency: in the dynamic environment in which the access patterns and information contents change rapidly, the parameters used in SCI will perform OLS with satisfactory service quality. As for the static environment in which the query profile and the database are updated infrequently, larger values of parameters are helpful to generate an optimized broadcast program, indicating the advantageous feature of MULS.

Index Terms—Mobile computing, sequential data broadcasting, ordered dependency, multilevel service quality, data broadcasting.

1 Introduction

Among various aspects in mobile computing technologies, data broadcasting has received the most attention in prior research. The broadcasting technique is a scalable way to disseminate the data such as stock prices, weather forecast, and traffic information from an *information system* to interested mobile clients. The physical bandwidth is partitioned into several logical channels. The server generates a *broadcast program* and broadcasts data items *periodically*. To retrieve the data on the air, mobile users should wait until items of interest appear in the channel. Such a technique is proposed in [2] and extended by Peng and Chen [25], Hsu et al. [11], Yee et al. [28], and so forth. Several advanced research topics such as the variant-bandwidth environment [29] and broadcasting heterogeneous items [14], [29] are discussed. Also, the concept of *on-demand* broadcasting is proposed in [5], [7] to dynamically disseminate the items without collecting the access patterns a priori.

Under many circumstances in a data broadcasting environment, a mobile user may be interested in multiple items simultaneously. For example, a mobile user interested in the stock information of one company may also request for the information of another relevant company. Therefore, the data items in a query (that is, the request sent from clients to the server for information services) are considered *dependent* on one another. To generate a broadcast program in such a *dependent data broadcasting* environment, the server should allocate the data items into single or multiple channels by collecting the query profile (that is, a table storing the dependency among the items for each query items) and the access probabilities. To provide better scheduling policies, a significant amount of research effort has been elaborated upon this aspect [9], [17], [19], [22], [26].

However, most of the prior works in *dependent data broadcasting* are based on an assumption that mobile users can access the items of interest in an arbitrary order. In fact, for many applications in the *dependent data broadcasting* environment, the data items should be accessed in a *sequential* order. For example, in the comics-by-phone services provided by Sony [1], a mobile user can read the comics in his/her mobile device. It is noted that each frame of the comics should be retrieved sequentially. As mentioned in [27], the SQL query sent by mobile users corresponds to a set of items with *ordered dependency*. Moreover, in [13], when hypertext objects are broadcast, since each object may contain text, graphics, video, and audio simultaneously, a request to one object will be transformed to a sequence of

- *H.-P. Hung is with the Graduate Institute of Communication Engineering, National Taiwan University, No. 1, Sec. 4, Roosevelt Road, Taipei, Taiwan, ROC. E-mail: hphung@arbor.ee.ntu.edu.tw.*
- *M.-S. Chen is with the Department of Electrical Engineering and the Graduate Institute of Communication Engineering, National Taiwan University, No. 1, Sec. 4, Roosevelt Road, Taipei, Taiwan, ROC. E-mail: mschen@cc.ee.ntu.edu.tw.*

Manuscript received 24 Mar. 2006; revised 17 Aug. 2006; accepted 9 Apr. 2007; published online 25 May 2007.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0135-0306.
Digital Object Identifier no. 10.1109/TKDE.2007.1067.

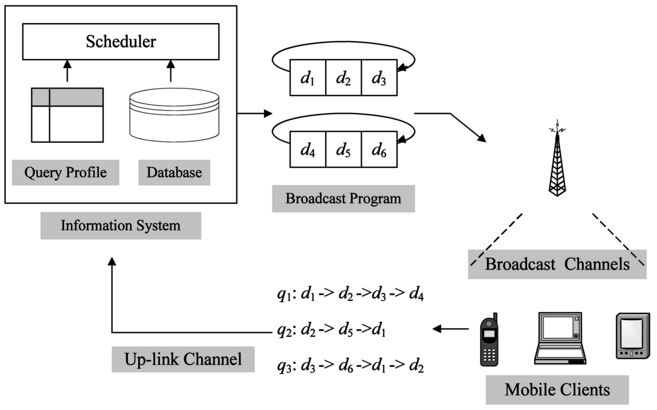

Fig. 1. The architecture of the information system for sequential data broadcasting.

subqueries. Therefore, these subqueries are also with ordered dependency. We discuss in this paper the scheduling algorithms of broadcasting items with *ordered dependency*. For simplicity, we use the term *sequential data broadcasting* to mean data broadcasting with *ordered dependency*.

1.1 Problem Formulation

The architecture of an information system for *sequential data broadcasting* is depicted in Fig. 1. Each query corresponds to a sequence of items. All of the queries are collected in the query profile at the server side. With the query profile and the database as the input, the information system will generate a broadcast program and broadcast the items periodically.[1] Due to the effect of periodicity, each item will appear repeatedly on the air. To identify the order of appearance, the items that appear simultaneously are conceptually located in the same *placement* of the broadcast program. The placement of a data item is defined as the identifier of the relative time slot at which this item is broadcast. When the server disseminates the items, the index structure will also be broadcast simultaneously. The index structure [8], [16] lies between time slots of broadcasting two consecutive items.[2] Once the index structure is downloaded, the client will be aware of the channel and the placement of each item. Therefore, he or she can switch from one channel to another to download all the items of interest. Moreover, to save the power of listening to the channel, if no item of interest is broadcast at a specific time slot, the client will enter doze mode until the next item of interest is broadcast.

To retrieve the items, a mobile user should wait until all the items of interest are downloaded sequentially. Since the items are broadcast via multiple channels, it is possible that more than one item of interest appears in different channels simultaneously. As shown in Fig. 1, since d_3 and d_6 are located in the same placement, they appear in different channels simultaneously. It is noted that the conventional data broadcasting schemes are based on the assumption that the data items are independent to each other. Therefore, the scheduling policy is only based on access probability. However, in sequential data broadcasting environments, in addition to access probabilities, the dependencies should be taken into account. For example, for mobile users issuing the query q_3, they should wait for one more cycle to retrieve d_6 after d_3 is downloaded. How to avoid these situations becomes an important issue in the sequential data broadcasting environments. By taking the features mentioned above into account, we should generate the broadcast program sophisticatedly so as to minimize the average access time of mobile users.

Theorem 1. *The problem of broadcasting data items with sequential relationship via multiple channels is NP-hard.*

Proof. We first introduce a similar problem of index allocation introduced in [21]. Consider an index tree structure in which the internal nodes $I = \{I_1, I_2, \ldots, I_m\}$ are viewed as the index nodes, whereas the leaf nodes $D = \{D_1, D_2, \ldots, D_n\}$ are viewed as the data nodes. When the index nodes and data nodes are broadcast via K channels, the data node should be received before all the index nodes are traversed from the root to the leaf. If the index nodes and the data nodes are allocated to the broadcast program, the quality of the broadcast program can be evaluated by

$$wait = \frac{\sum_{D_i \in D} W(D_i)T(D_i)}{\sum_{D_i \in D} W(D_i)},$$

where $W(D_i)$ represents the weighting factor at each data node, and $T(D_i)$ denotes the offset in terms of time slots between the root node and the leaf node in the broadcast program. The index allocation problem is to find a broadcast program in such a way that the value of $wait$ can be minimized. Note that this problem has been proved to be NP-hard in [21].

In our sequential data broadcasting problem, we aim at minimizing the average waiting time according to the sequential relationship in the query profile. Consider a special case in which the sequential relationship of the data items in the query profile forms a tree structure,[3] that is, all of the queries have some prefix items in common. Under this circumstance, our sequential data broadcasting problem can be reduced to the index allocation problem. Therefore, the problem of sequential data broadcasting can also be viewed as an NP-hard problem. □

1.2 Our Solution

We propose in this paper a general framework of the scheduler named MULS (standing for MUltiLevel Service) for an information system to provide multilevel service

1. Although on-demand broadcasting is also a practical solution to disseminating data, under several circumstances (for example, low-power status), the users would like to receive the data from periodically broadcasting channels without sending the requests. Therefore, we consider in this paper a general environment that the data items are broadcast periodically.

2. Typically, an index structure can be viewed as a list in which each element is a mapping from the item identifier to the channel identifier. The length of the list is equal to the number of broadcast items. Note that this list can also be divided into several sublists, which are broadcast between different time slots so as to reduce the traffic load in the channel. If the effect of index broadcasting is considered, the average access time will be offset by a small value since the size of identifier is much smaller than that of item content.

3. In fact, the sequential relationship in a general query profile may form any directed graph.

quality in the *sequential data broadcasting* environment. MULS contains two primary stages. The first stage is the online scheduling (OLS), whereas the second stage is the optimization procedure. In the first stage, we propose an OLS algorithm to allocate the data items into multiple channels. As for the second stage, we devise an optimization procedure, called Sampling with Controlled Iteration (SCI), to enhance the quality of broadcast programs generated by algorithm OLS. By tuning the control parameters of procedure SCI, the MULS framework is able to provide different service qualities by compromising the effectiveness and efficiency in different dissemination environments.

Explicitly, with the database and the query profile as the input, the MULS framework can first efficiently generate a broadcast program with satisfactory quality using algorithm OLS. Algorithm OLS is designed from the viewpoint of *placement*. Given two consecutive items in a query, algorithm OLS will schedule the data in either *forward* direction or *backward* direction so that the *distance* from the placement of one item to that of the other is minimized. Physically, in a specific broadcast program, the *distance* represents the number of items that a mobile user has to probe before the next item of interest appears on the air. Therefore, the time that a mobile user spends probing the items should be in proportion to the amount of *distance*. Algorithm OLS is very efficient in performing OLS by updating the broadcast program dynamically.

In order to optimize the broadcast program generated by algorithm OLS, we devise procedure SCI in the MULS framework to enhance the quality and provide flexibility. We first propose procedure BASIC for optimization, which will be used for comparison purposes with SCI. Procedure BASIC can be executed iteratively. In each iteration, BASIC searches all possible exchanging operations of two items located in different placements and determines the best choice. At the end of the iteration, the broadcast program will be updated according to the best exchanging operation. Procedure BASIC continues until the *local optimum* is reached, where no exchanging operation causes the reduction of the average access time. Based on the concept of procedure BASIC, three variants, that is, sample-based optimization (SAMPLE), iteration-based optimization (ITERATION), and SCI, can be used to enhance the overall flexibility. Instead of inspecting all possible exchanging operations, procedure SAMPLE, only selects a predetermined number of samples to reduce the average access time. It is noted that SAMPLE can also reach the local optimum. On the other hand, ITERATION is a parameterized algorithm extended from BASIC. With a parameter controlling the number of iterations, procedure ITERATION can provide a significant enhancement. The final procedure SCI, which integrates the advantageous features of SAMPLE and ITERATION, is designed to attain the flexibility required in various environments.

In fact, the broadcast program generated by the MULS framework can be compatible with the conventional dependent data broadcasting environment in which the items are retrieved in an arbitrary order since the sequential data broadcasting environment has stricter scheduling rules. In order to validate the proposed MULS framework, several experiments are conducted. The experimental results show that algorithm OLS outperforms the conventional approaches and achieves satisfactory quality, and the proposed optimization procedures can further enhance the broadcast program significantly. Moreover, from the performance analysis among all optimization procedures, SCI is shown to be the most scalable so that it can still perform well for broadcasting a large-scale database. Therefore, to provide multilevel service quality for different requirements, the MULS framework, which combines the benefits of OLS and SCI, emerges as a general and powerful framework. In the dynamic environment in which the access patterns and information contents change rapidly, the information system should regenerate the broadcast program frequently once the current broadcast program cannot satisfy the requirements of mobile users. Under this circumstance, the parameters x and y used in SCI can be tuned small for OLS. As for the static environment in which the query profile and the database are updated infrequently, the broadcast program will remain the same over time. Larger values of x and y are helpful to generate an offline broadcast program, which indicates the advantageous feature of MULS.

1.3 Outline

The rest of this paper is outlined as follows: In Section 2, preliminaries will be given. In Section 3, we will present the design of the MULS framework. The experimental results will be shown in Section 4 and, finally, this paper concludes in Section 5.

2 Preliminaries

2.1 Related Works

In the dependent data broadcasting environment, unlike the conventional data broadcasting, the information system is able to deal with the queries that may contain multiple items simultaneously. There is no restriction on the order of downloading the dependent data. To schedule the items, several algorithms are also developed. Chung and Kim [9], [10] used the *QueryDistance* to measure the degree of coherence for the data set accessed by a query. Physically, the *QueryDistance* represents the minimal number of items that a user should probe since the first item of interest is accessed until the last item is downloaded. The Query Expansion Method (QEM) algorithm can be viewed as a *greedy* algorithm since it will choose the most profitable position to *append* the items in a query. According to Chung and Kim's analytical model, Lee and Lo [19] enhanced the performance of algorithm QEM by loosening the restrictions in [9], [10]. The major contributions of Lee and Lo's work [19] are listed as follows: 1) The *query selection problem* (that is, choosing the most profitable queries to broadcast) is proved to be NP-hard. 2) An extended algorithm named Modified Query Expansion Method (MQEM) is proposed to enhance algorithm QEM. 3) A new *data-oriented* approach to compete with MQEM was designed according to the *data access graph*, which is an undirected graph to identify the dependent relationship among the data items in different queries. Recently, Hung et al. [15] also proposed a greedy algorithm, Placement-Based Allocation (PBA), in scheduling dependent items via multiple channels. Algorithm PBA

TABLE 1
Description of the Symbols

Symbols	Descriptions
N	Number of data items
D	The broadcast database
Q	The query profile
K	Number of channels
L	Length of the broadcast cycle (Number of the placements)
d_i	The i-th item in D
p_i	The placement of d_i
q_i	The i-th query in Q
s	Size of each data item
B	Bandwidth of each broadcast channel

TABLE 2
Description of the Acronyms

Acronyms	Descriptions
MULS	A framework providing multi-level service
OLS	An on-line scheduling algorithm
BASIC	An optimization procedure providing basic optimization
SAMPLE	A sample-based optimization procedure
ITERATION	An iteration-based optimization procedure
SCI	An optimization procedure which integrates SAMPLE and ITERATION

generates a broadcast program from the viewpoint of the *placement*. Since each query may contain multiple items in a dependent data broadcasting environment, the scheduling policy of PBA is to avoid the occurrence of *conflicting items*, defined as the situation that multiple items in a query are located in the same placement. Moreover, algorithm PBA allows the query with higher access probability to have a higher priority to schedule its items. Such provision can effectively reduce the impact of *conflicting items*.

In sequential data broadcasting, also known as broadcasting with *ordered dependency*, the access to items should follow a sequential order. In [12], Hu and Chen addressed the issue of scheduling sequential items in an infinite time horizon via a single channel and transformed the scheduling problem as a problem of finding a solution in solving the algebraic problem. As for broadcasting sequential items via multiple channels, in [13], Huang and Chen formulated the analytical model for *sequential data broadcasting* and proposed a solution, which is based on the genetic algorithm (GA). Initially, a set of broadcast programs are generated randomly and encoded as *chromosomes*. Each chromosome contains a numerical attribute called *fitness*. The larger value of fitness indicates that the corresponding broadcast program can achieve the shorter average access time. In each generation, the chromosomes evolve via several operations such as *crossover* and *mutation*. The evolution procedure continues until either all chromosomes in the population have the same performance or a predetermined bound that restricts the number of generations is reached. Recently, in [20], Liu and Lin claimed that the sequential relationship of items throughout the query profile can be modeled as a directed graph and proposed two algorithms: Revised Topological Sort (RTS) and Minimum Offset Algorithm (MOA). Algorithm RTS assigns a higher scheduling priority to the edge with a higher aggregated weight. On the other hand, algorithm MOS schedules the items based on two steps: 1) assigning level and 2) partitioning with minimum offset. In step 1, each vertex corresponding to a broadcast item will be assigned a level number according to the sequential relationship. In step 2, the final placement of an item will be determined according to the level number of the corresponding vertex.

2.2 Analytical Models of Sequential Data Broadcasting

Without loss of generality, we consider that the database D with size $|D| = N$ contains all the data items that mobile clients may query.[4] Let the query profile Q represent the collection of all distinct queries that mobile users may issue. Each query $q_i \in Q$ is regarded as a request for multiple items that should be retrieved in a sequential order. Given the fixed number of channels K, all the items in D are distributed evenly into K channels in a broadcast program. Therefore, each channel has the same broadcast cycle L. That is, $L = \lceil \frac{N}{K} \rceil$. L is also viewed as the number of *placements* in the broadcast program. For each $d_i \in D$, its placement is denoted by p_i. Table 1 lists the related symbols and the corresponding descriptions. Moreover, Table 2 also lists the acronyms throughout this paper. To facilitate the description of the MULS framework, we employ the same notation and definitions as in [13].

Definition 1. *Consider two consecutive items d_i and d_j in a specific query. To measure the waiting time of downloading the two items in a broadcast program, the function $dst(i,j)$ is defined as the* distance *from the placement of d_i to the placement of d_j. That is,*

$$dst(i,j) = \begin{cases} p_j - p_i - 1, & \text{if } i \neq j \text{ and } p_j > p_i, \\ L - p_i + p_j - 1, & \text{if } i \neq j \text{ and } p_j \leq p_i, \\ 0 & \text{if } i = j. \end{cases}$$

Definition 2. *The average access time of a query q_i, denoted by $T_A(q_i)$, is defined as the average time that a mobile user should spend accessing all the items in query q_i.*

Note that $T_A(q_i)$ can be decomposed into three parts: the start-up time, the waiting time, and the retrieval time. The start-up time $T_S(q_i)$ represents the duration since a mobile user issues query q_i until the first item of interest appears in one of the channels. The waiting time $T_W(q_i)$ is defined as the summation of the time intervals between the moment that the mobile user completes the retrieval of the current item in q_i and the moment that the mobile device starts to retrieve the next item in q_i. $T_W(q_i)$ can be viewed as the aggregated time of probing the items, which reflect the waiting time of probing until the item of interest appears. The retrieval time $T_R(q_i)$ is the aggregated time, whereas the mobile device indeed reads data items from broadcast channels. Therefore, $T_A(q_i)$ can be formulated as $T_A(q_i) = T_S(q_i) + T_W(q_i) + T_R(q_i)$, where

4. In a large-scale dissemination system, the items that mobile users are interested in may not be immediately available. Such a *data-staging* problem can be resolved according to the approaches proposed in [6].

$$T_S(q_i) = \frac{L}{2} \times \frac{s}{B}, T_R(q_i) = |q_i| \times \frac{s}{B},$$
$$T_W(q_i) = \left[\sum_{k=1}^{|q_i|-1} dst(q_i(k), q_i(k+1))\right] \times \frac{s}{B}.$$

Note that $q_i(k)$ identifies the kth item in q_i; s and B denote the item size and channel bandwidth, respectively.[5]

Definition 3. *Let $Pr(q_i)$ represent the access probability of the query q_i. The average access time of a query profile Q, denoted by $T_A(Q)$, is defined as the average time that a mobile user should spend accessing a query in Q. $T_A(Q)$ can be formulated as*

$$\begin{aligned} &T_A(Q) \\ &= \sum_{i=1}^{|Q|}[T_A(q_i) \times Pr(q_i)] \\ &= \sum_{i=1}^{|Q|} T_S(q_i)Pr(q_i) + \sum_{i=1}^{|Q|} T_W(q_i)Pr(q_i) + \sum_{i=1}^{|Q|} T_R(q_i)Pr(q_i). \end{aligned} \tag{1}$$

If we define

$$T_S(Q) = \sum_{i=1}^{|Q|} T_S(q_i)Pr(q_i),$$
$$T_R(Q) = \sum_{i=1}^{|Q|} T_R(q_i)Pr(q_i), \text{ and}$$
$$T_W(Q) = \sum_{i=1}^{|Q|} T_W(q_i)Pr(q_i),$$

(1) can be rewritten as

$$T_A(Q) = T_S(Q) + T_R(Q) + T_W(Q), \tag{2}$$

where

$$\begin{aligned} T_S(Q) &= \frac{L}{2} \times \frac{s}{B}, T_R(Q) = \left\{\sum_{i=1}^{|Q|} |q_i| \times Pr(q_i)\right\} \times \frac{s}{B}, \\ T_W(Q) &= \sum_{i=1}^{|Q|} \sum_{j=1}^{|q_i|-1} [dst(q_i(j), q_i(j+1)) \times Pr(q_i)] \times \frac{s}{B}. \end{aligned} \tag{3}$$

Definition 4. *An auxiliary table (referred to as table a) of a query profile Q is a $|D|$ by $|D|$ array, where each entry $a(x, y)$ is defined as*

$$\sum_{i=1}^{|Q|} [\text{number of occurrences of the pair } (d_x \rightarrow d_y) \text{ in } q_i] \times Pr(q_i).$$

5. We focus in this paper on improving the service quality of the existing approach based on the assumption of constant bandwidth. However, our work (that is, the proposed scheduling algorithms) can also be suitable for the model of variant bandwidth. It is recognized that in a real mobile computing environment, the bandwidth is likely to be shared among all available channels. In [29], the authors derive the analytical model of broadcasting nonsequential items in a variant-bandwidth environment. To broadcast sequential items in a variant-bandwidth environment, the analytical model and the MULS framework can be extended from [29].

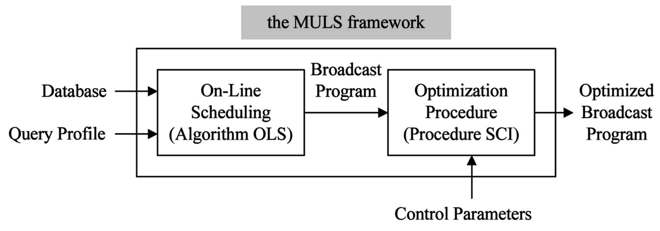
the MULS framework
Database
Query Profile
On-Line Scheduling (Algorithm OLS)
Broadcast Program
Optimization Procedure (Procedure SCI)
Optimized Broadcast Program
Control Parameters

Fig. 2. Architecture of the MULS framework.

With the table a of the query profile Q, $T_W(Q)$ can further be rewritten as

$$T_W(Q) = \sum_{j=1}^{|D|} \sum_{k=1}^{|D|} [a(j,k) \times dst(j,k)] \times \frac{s}{B}. \tag{4}$$

3 Design of the MULS Framework for Multilevel Service Quality

In this section, we will present the design of the MULS framework. In Section 3.1, an overview of the MULS framework will be given. In Section 3.2, an OLS algorithm will be proposed. We will describe the optimization procedure SCI in Section 3.3. Finally, several implementation issues will be discussed in Section 3.4.

3.1 Overview

Fig. 2 depicts the architecture of the proposed MULS framework for multilevel service quality in the sequential data broadcasting environment. The framework of the scheduler contains two primary stages. With the database and the query profile as the input,[6] the first stage can efficiently generate a broadcast program with satisfactory quality. After that, the broadcast program will be optimized in the second stage. The optimization procedure allows single or multiple parameters to strike a compromise between effectiveness and efficiency. The information system can thus provide multiple levels of service quality by tuning the control parameters. In the MULS framework, there are two primary factors: x and y. The value of x, which determines the number of samples, depends on the capability of the information system. Usually when the information system has high computing power, we will use a larger x to enhance the service quality. On the other hand, the value of y, which determines the number of iterations, depends on the dynamics in the access patterns. In the dynamic environment, in which the distribution of access patterns varies rapidly, we will use a smaller y to speed up the update frequency of the broadcast program.

3.2 Online Scheduling

The OLS algorithm is designed in such a way that *all* the data items in D can be effectively allocated in the broadcast program within an almost linear complexity. To achieve this goal, we design algorithm OLS. The algorithmic form of algorithm OLS and the relevant functions are outlined in Fig. 3 and Fig. 4, respectively.

6. The query profile and the broadcast database may change dynamically. The query profile will be updated once a new query is received. Moreover, a sliding window will be employed so as to maintain the most recent access patterns of mobile clients.

```
Algorithm OLS
Input: sorted query profile $Q$, database $D$, number of channels $K$
/* the query in Q are sorted in descending order according to the access probabilities */
Output: the broadcast program $P$
1  Set $L = \left\lceil \frac{|D|}{K} \right\rceil$
2  Create an empty broadcast program $P$, where $P = \{D_l \mid 1 \le l \le L\}$, each $D_i$ is an empty set
3  for $i$ = 1 to $|Q|$
4      for $j$ = 1 to $|q_i|$-1
5          if $d_{q_i(j)}$ is scheduled and $d_{q_i(j+1)}$ is unscheduled
6              $P$=FSchedule($d_{q_i(j)}$, $d_{q_i(j+1)}$, $P$)
7          else if $d_{q_i(j)}$ is unscheduled and $d_{q_i(j+1)}$ is scheduled
8              $P$=BSchedule($d_{q_i(j)}$, $d_{q_i(j+1)}$, $P$)
9          else if $d_{q_i(j)}$ and $d_{q_i(j+1)}$ are both unscheduled
10             $P$=FBSchedule($d_{q_i(j)}$, $d_{q_i(j+1)}$, $P$)
11         end if
12     end for
13     if all data items in $D$ are scheduled
14         break
15     end if
16 return $P$
```

Fig. 3. Algorithm OLS.

```
Function FSchedule
Input: the $j$-th item and the $j$+1-th item in $q_i$ (i.e., $d_{q_i(j)}$ and $d_{q_i(j+1)}$)
       the temporary broadcast program $P$
Output: the updated broadcast program
1  From the broadcast program $P$, find out the placement of $d_{q_i(j)}$ (i.e., $p_{q_i(j)}$ )
2  Find out a legal placement $p_x$ in such a way that $dst(q_i(j), x)$ is minimized
3  Insert $d_{q_i(j+1)}$ into $p_x$
4  Return $P$
```

```
Function BSchedule
Input: the $j$-th item and the $j$+1-th item in $q_i$ (i.e., $d_{q_i(j)}$ and $d_{q_i(j+1)}$)
       the temporary broadcast program $P$
Output: the updated broadcast program
1  From the broadcast program $P$, find out the placement of $d_{q_i(j+1)}$ (i.e., $p_{q_i(j+1)}$)
2  Find out a legal placement $p_x$ in such a way that $dst(x, q_i(j+1))$ is minimized
3  Insert $d_{q_i(j+1)}$ into $p_x$
4  Return $P$
```

```
Function FBSchedule
Input: the $j$-th item and the $j$+1-th item in $q_i$ (i.e., $d_{q_i(j)}$ and $d_{q_i(j+1)}$)
       the temporary broadcast program $P$
Output: the updated broadcast program
1  Let MinPlace = argmin$\{|D_i|, 1 \le i \le L\}$ /* the placement with fewest items */
2  Put $d_{q_i(j)}$ into MinPlace
3  $P$=FSchedule($d_{q_i(j)}$, $d_{q_i(j+1)}$, $P$)
4  Return $P$
```

Fig. 4. Relevant functions in algorithm OLS.

According to (2), we observe that $T_S(Q)$ and $T_R(Q)$ are irrelevant to the broadcast program generation since B and s are constants and L, $|q_i|$, and $Pr(q_i)$ are determined regardless of the scheduling policy. That is, the quality of a broadcast program only influences the amount of $T_W(Q)$. Therefore, given a query q_i, algorithm OLS aims to minimize $T_W(Q)$ by reducing the *distance* between $q_i(j)$ and $q_i(j+1)$ for each $j < |q_i|$. There are two basic intuitions in algorithm OLS: 1) the items appearing in popular queries deserve a higher priority to be scheduled, and 2) the items with a higher priority have better chances to minimize the distance. To schedule the items in D, a broadcast program P, which contains L empty sets $\{D_l, 1 \le l \le L\}$, is first created. Note that each set D_l corresponds to the items located in placement l. The scheduling procedure can be viewed as filling all items into L sets in such a way that each set should contain no more than K items. When the items in each query q_i are processed, we use $d_{q_i(j)}$ and $d_{q_i(j+1)}$ to identify the jth and the $(j+1)$th item in q_i. According to the status of $d_{q_i(j)}$ and $d_{q_i(j+1)}$, there are three scenarios for algorithm OLS to allocate $d_{q_i(j)}$ and $d_{q_i(j+1)}$ into P:

1. $d_{q_i(j)}$ is scheduled, but $d_{q_i(j+1)}$ is unscheduled. We devise a procedure $FSchedule$, standing for *forward scheduling*, to determine the placement where $d_{q_i(j+1)}$ should be put according to the placement of $d_{q_i(j)}$.
2. $d_{q_i(j)}$ is unscheduled, but $d_{q_i(j+1)}$ is scheduled. We use procedure $BSchedule$, standing for *backward scheduling*, to determine the item set in which $d_{q_i(j)}$ should be put according to the placement of $d_{q_i(j+1)}$.
3. Both $d_{q_i(j)}$ and $d_{q_i(j+1)}$ are unscheduled. We use procedure $FBSchedule$. The item $d_{q_i(j)}$ will be put in the set with fewest items. After that, we use procedure $FSchedule$ to determine the placement of $d_{q_i(j+1)}$. In $FBSchedule$, exchanging the scheduling order of $d(j)$ and $d(j+1)$ will not affect the access time of the current query. However, it may change the access time of the successive queries that contain $d(j)$ or $d(j+1)$. Since algorithm OLS is greedy in nature and should be executed within almost linear time, we will ignore the effect of successive queries.

During the execution of algorithm OLS, we employ three procedures $FSchedule$, $BSchedule$, and $FBSchedule$ to allocate an item into a placement so as to minimize the distance between two consecutive items in a query. Since $FBSchedule$ is based on $FSchedule$. We only discuss $FSchedule$ and $BSchedule$. Given two consecutive items $d_i \rightarrow d_j$, $FSchedule$ (respectively, $BSchedule$) will put d_j (respectively, d_i) in a $legal$ placement so that $dst(i,j)$ is minimized. Note that a placement l is $legal$ iff $|D_l| < K$. For popular queries, the items have better chances to be scheduled in the placement where $dst(i,j)$ is minimized. On the other hand, when the items of unpopular queries are processed, the placement with the minimum $dst(i,j)$ may conflict with legality. The two procedures will put the items into inferior but legal placements.

Lemma 1. *The complexity of algorithm OLS is* $O(|Q| \log |Q|) + O(N \log N)$ *when* N *data items should be scheduled.*

Proof.

1. Initially, it costs $O(|Q| \log |Q|)$ to sort the queries according to the access probabilities.
2. Next, consider the cost of scanning. Since the scheduler does not allow each data item to appear more than once in the broadcast program, the items already existing in the broadcast program will be skipped during the scan. Therefore, the complexity of scanning the broadcast profile will be $O(N)$.
3. To allocate each item, we consider $FBSchedule$, which causes the highest complexity in scheduling the items. Before all the items are scheduled, we can build a heap to store all the placements. The placements are sorted according to the number of items contained. When $FBSchedule$ is executed, the heap will return the placement in the front for allocating the items. After that, it costs $O(\log L)$ to update the heap. Since $L = N/K$, it costs $O(\log N)$ to allocate each item.

query	Prob	contents
q_1	0.310	d_6 -> d_9 -> d_4 -> d_5
q_2	0.178	d_3 -> d_6 -> d_7 -> d_4 -> d_2 -> d_8 -> d_{11}
q_3	0.128	d_3 -> d_5 -> d_6 -> d_9
q_4	0.102	d_5 -> d_6 -> d_4
q_5	0.085	d_9 -> d_5 -> d_{10}
q_6	0.074	d_1 -> d_8 -> d_{11} -> d_6
q_7	0.065	d_7 -> d_2 -> d_4 -> d_3 -> d_9 -> d_{12}
q_8	0.058	d_7 -> d_1 -> d_8 -> d_{11} -> d_6 -> d_9 -> d_4

(a)

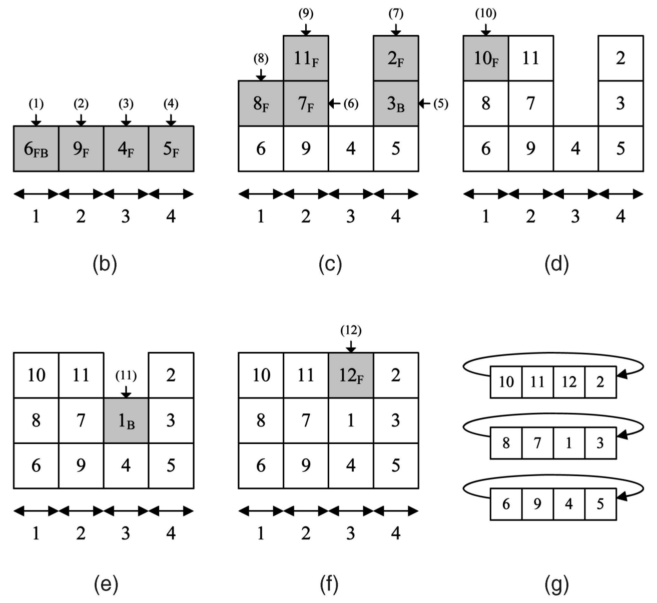

Fig. 5. Example of algorithm OLS. (a) Query profile of the example. (b) Scheduling items in q_1. (c) Scheduling items in q_2. (d) Scheduling items in q_5. (e) Scheduling items in q_6. (f) Scheduling items in q_7. (g) Broadcast program of OLS.

4. According to items 1, 2, and 3 of this proof, Lemma 1 can thus be proved. □

Lemma 2. *Algorithm OLS is* $(|Q|-1)$*-competitive compared to the optimal solution.*

Proof. We consider the worst case in which the access probability of each query is the same (that is, $Pr(q_i)=\frac{1}{|Q|}$ for $1\le i\le|Q|$), and $K=1$. Moreover, in the worst case, we have $q_1=\{d_1\rightarrow d_2\rightarrow d_3\rightarrow\ldots\rightarrow d_N\}$ and $q_i=\{d_N\rightarrow d_{N-1}\rightarrow d_{N-2}\rightarrow\ldots\rightarrow d_1\}$ for $2\le i\le|Q|$.[7] The broadcast program generated by algorithm OLS, denoted by P_{OLS}, will be $P_{OLS}=\{\{d_1\},\{d_2\},\{d_3\},\ldots,\{d_N\}\}$, whereas the optimal broadcast program, denoted by P_{OPT}, will be $P_{OLS}=\{\{d_N\},\{d_{N-1}\},\{d_{N-2}\},\ldots,\{d_1\}\}$. Note that optimal broadcasting refers to the broadcast program that leads to the minimum average access time among all possibilities for this profile. To evaluate the quality of the broadcast program, we only consider $T_W(Q)$, since $T_A(Q)$ and $T_S(Q)$ remain the same regardless of the broadcast programs. Let $T_W^{(OLS)}$ and $T_W^{(OPT)}$ be the average waiting time of the broadcast program generated by algorithm OLS and the optimal broadcast program, respectively. According to (3), $T_W^{(OLS)}$ and $T_W^{(OPT)}$ can be formulated as

$$\begin{aligned}T_W^{(OLS)} &= 0+(|Q|-1)*(N-2)*\frac{1}{|Q|}*\frac{s}{B},\\ T_W^{(OPT)} &= (|D|-2)*\frac{1}{|Q|}*\frac{s}{B}+(|Q|-1)*0.\end{aligned}$$

In the worst case, we can obtain

$$T_W^{(OLS)}/T_W^{(OPT)}=(|Q|-1).$$

Therefore, algorithm OLS is $(|Q|-1)$-competitive compared to the optimal solution. □

We can simply compare algorithm OLS to the conventional RTS [20] approach in this worst case. It is noted that in algorithm RTS, the sequential pattern $x->y$ can be viewed as the edge from x to y, and the weight of this edge can be obtained by aggregating the occurrence probability of $x->y$ in the query profile. When RTS is executed, all of the edges will be sorted according to the weight. The edge with a higher weight will have a higher priority to be processed. Compared to algorithm OLS, since at most N^2 edges have been sorted, the complexity of RTS will be at least $O(N^2\log N)$, which is larger than the scheduling cost of algorithm OLS. As for the quality of the broadcast program, as will be shown in the experimental results, in general data broadcasting environments in the presence of highly skewed access patterns, algorithm OLS will have a better performance than RTS.

Example 1. An execution scenario of algorithm OLS is presented in Fig. 5. The query profile is shown in Fig. 5a. Consider eight queries sorted according to the access probability in descending order. Algorithm OLS will generate a broadcast program to schedule 12 data items into three channels. Initially, OLS generates an empty broadcast program with cycle length $L=4$. Thus, we use $1\sim 4$ to label the placements. The notation D_l, $1\le l\le 4$, is used to represent the set of items put in the lth placement. Since q_1 has the highest access probability, the items in q_1 have the highest priority to be scheduled. During the execution of algorithm OLS, we use the subscripts to identify which scheduling scheme is employed. Moreover, the numbers in brackets will indicate the order of scheduling items. Fig. 5b shows the result of scheduling items in q_1. Each item is scheduled so that $dst(i,j)$ for each d_i, d_j in q_1 can be

7. It is noted that in our OLS algorithm, the query with the highest access probability will have the highest priority to be processed. Therefore, in the worst case, the priorities of the queries are ambiguous, that is, each query has the same access probability. Moreover, in the worst case, the scheduling result of a query will be the inverse of its sequential relationship. Therefore, our example is the worst case of scheduling N items in $|Q|$ queries.

minimized. To determine the placement in which the first item d_6 should be put, algorithm OLS will select the placement corresponding to the set with fewest items. Note that only the procedure $FSchedule$ is used when the items in q_1 are scheduled. In Fig. 5c, the items in q_2 are put in the broadcast program P according to the scheduling result in Fig. 5b. Note that the procedure $BSchedule$ is executed to determine the placement of d_3 since its next queried item, d_6, appears in D_1. As shown in Fig. 5e, when algorithm OLS proceeds to determine the placement of d_1, procedure $BSchedule$ only selects a legal placement, that is, placement 3, since the best placement, that is, placement 4, is occupied with three items. Once all the data items are scheduled, as shown in Fig. 5f, algorithm OLS will complete the scheduling procedure and return the broadcast program P, as shown in Fig. 5g.

3.3 Sampling with Controlled Iteration

Although algorithm OLS can be executed very efficiently, it only achieves a satisfactory but not optimal solution. The reason is explained as follows: When algorithm OLS tries to determine the placement of a specific item, the most popular query containing the item is considered and the effect of other unpopular queries is neglected. To optimize the broadcast program generated by algorithm OLS, we should take the effect contributed by *all* queries into account. The following theorem and corollaries are listed to facilitate the description.

Definition 5. *The operator $\odot$ is defined as the modulus operator, indicating that $x \odot y = x - ny$, where n is the floor value of x/y if $y \neq 0$. For example, $6 \odot 5 = 1$; $(-1) \odot 5 = 4$.*[8]

Lemma 3. $dst(i,j) = -[(p_i - p_j) \odot L] + L - 1$ *for* $i \neq j$.

Proof. First, consider $p_j > p_i$. We have

$$\begin{aligned} dst(i,j) &= p_j - p_i - 1 \\ &= -(p_i - p_j + L) + L - 1 \\ &= -[(p_i - p_j) \odot L] + L - 1. \end{aligned}$$

As for $p_j \leq p_i$, we have

$$\begin{aligned} dst(i,j) &= p_j - p_i + L - 1 \\ &= -(p_i - p_j) + L - 1 \\ &= -[(p_i - p_j) \odot L] + L - 1. \end{aligned}$$

□

Corollary 3.1.

$$dst(i_1,j_1) - dst(i_2,j_2) = (p_{i_2} - p_{j_2}) \odot L - (p_{i_1} - p_{j_1}) \odot L.$$

Corollary 3.2. $dst(i,j) - dst(j,i) = 2[(p_j - p_i) \odot L] - L$ *for* $p_i \neq p_j$.

As presented in Section 3.2, the quality of a broadcast program only influences the amount of $T_W(Q)$. In (4), since s and B are constants, we use a cost function to simplify $T_W(Q)$. The cost is defined as $C = \sum_{i=1}^{|D|} \sum_{j=1}^{|D|} a(i,j) dst(i,j)$. Next, we consider the cost reduction of exchanging two items in different placements. The cost before exchanging the placements of two items d_u and d_v, denoted by $C_B(u,v)$, is formulated as follows:

$$\begin{aligned} &C_B(u,v) \\ &= \sum_{i=1}^{|D|} \sum_{j=1}^{|D|} a(i,j) dst(i,j) \\ &= \sum_{\substack{i=1 \\ i \neq u,v}}^{|D|} \sum_{\substack{j=1, \\ j \neq u,v}}^{|D|} a(i,j) dst(i,j) + \sum_{\substack{j=1, \\ j \neq u,v}}^{|D|} a(u,j) dst(u,j) + \\ &\sum_{\substack{j=1, \\ j \neq u,v}}^{|D|} a(v,j) dst(v,j) + \sum_{\substack{i=1, \\ i \neq u,v}}^{|D|} a(i,u) dst(i,u) + \\ &\sum_{\substack{i=1, \\ i \neq u,v}}^{|D|} a(i,v) dst(i,v) + a(u,v) dst(u,v) + a(v,u) dst(v,u). \end{aligned} \tag{5}$$

We also formulate $C_A(u,v)$, that is, the cost after exchanging the placements of d_u and d_v, as

$$\begin{aligned} &C_A(u,v) \\ &= \sum_{\substack{i=1, \\ i \neq u,v}}^{|D|} \sum_{\substack{j=1, \\ j \neq u,v}}^{|D|} a(i,j) dst(i,j) + \sum_{\substack{j=1, \\ j \neq u,v}}^{|D|} a(u,j) dst(v,j) + \\ &\sum_{\substack{j=1, \\ j \neq u,v}}^{|D|} a(v,j) dst(u,j) + \sum_{\substack{i=1, \\ j \neq u,v}}^{|D|} a(i,u) dst(i,v) + \\ &\sum_{\substack{i=1, \\ j \neq u,v}}^{|D|} a(i,v) dst(i,u) + a(u,v) dst(v,u) + a(v,u) dst(u,v). \end{aligned} \tag{6}$$

Defining the cost reduction $\Delta C(u,v)$ as

$$\Delta C(u,v) = C_B - C_A,$$

we can derive the cost reduction according to (5) and (6):

$$\begin{aligned} &\Delta C(u,v) \\ &= \sum_{j=1, j \neq u,v}^{|D|} a(u,j)[(p_v - p_j) \odot L - (p_u - p_j) \odot L] + \\ &\sum_{j=1, j \neq u,v}^{|D|} a(v,j)[(p_u - p_j) \odot L - (p_v - p_j) \odot L] + \\ &\sum_{i=1, j \neq u,v}^{|D|} a(i,u)[(p_i - p_v) \odot L - (p_i - p_u) \odot L] + \\ &\sum_{i=1, i \neq u,v}^{|D|} a(i,v)[(p_i - p_u) \odot L - (p_i - p_v) \odot L] + \\ &a(u,v)\{2[(p_v - p_u) \odot L] - L\} + \\ &a(v,u)\{2[(p_u - p_v) \odot L] - L\}. \end{aligned} \tag{7}$$

To obtain the cost reduction of an exchanging operation, an exhaustive way is to calculate $C_B(u,v)$ and $C_A(u,v)$ by definition. This approach requires $O(N^2)$ complexity. Since there are redundant calculations in the exhaustive approach,

8. The modulus definition in this paper is different from the operator % in C and Java language but the same as the function $\mathrm{mod}(x,y)$ in Matlab.

```
Procedure BASIC
Input: original broadcast program $P$
Output: optimized broadcast program $P'$
while true
    $\Delta C_{Max} = 0$, $d_{Max1}$=null, $d_{Max2}$=null
    for $i_1$=1 to $N$-1
        for $i_2$=$i_1$+1 to $N$
            if $p_{i1} \neq p_{i2}$
                calculate $\Delta C(i_1,i_2)$
                if $\Delta C_{Max} < \Delta C(i_1,i_2)$
                    $\Delta C_{Max} = \Delta C(i_1,i_2)$
                    $d_{Max1}= d_{i1}$, $d_{Max2}= d_{i2}$
    if $\Delta C_{Max} == 0$
        break while
    else
        exchange the placements of $d_{Max1}$ and $d_{Max2}$
return $P$
```

(a)

```
Procedure SAMPLE
Input: original broadcast program $P$, parameter $x$
Output: optimized broadcast program $P'$
while true
    $\Delta C_{Max} = 0$, $d_{Max1}$=null, $d_{Max2}$=null, x_count=0
    Create a set $S$ storing all possible exchanging pairs
    while $|S| > 0$
        Randomly select a pair $(i_1, i_2)$ from $S$
        Remove $(i_1, i_2)$ from $S$
        if $p_{i1} \neq p_{i2}$
            calculate $\Delta C(i_1,i_2)$
            if $\Delta C_{Max} < \Delta C(i_1,i_2)$
                $\Delta C_{Max} = \Delta C(i_1,i_2)$, x_count ++
                $d_{Max1}= d_{i1}$, $d_{Max2}= d_{i2}$
        if x_count == $x$
            break while
    if $\Delta C_{Max} == 0$
        break while
    else
        exchange the placements of $d_{Max1}$ and $d_{Max2}$
return $P$
```

(b)

```
Procedure ITERATION
Input: original broadcast program $P$, parameter $y$
Output: optimized broadcast program $P'$
for y_count = 1 to $y$
    $\Delta C_{Max} = 0$, $d_{Max1}$=null, $d_{Max2}$=null
    for $i_1$=1 to $N$-1
        for $i_2$=$i_1$+1 to $N$
            if $p_{i1} \neq p_{i2}$
                calculate $\Delta C(i_1,i_2)$
                if $\Delta C_{Max} < \Delta C(i_1,i_2)$
                    $\Delta C_{Max} = \Delta C(i_1,i_2)$
                    $d_{Max1}= d_{i1}$, $d_{Max2}= d_{i2}$
    if $\Delta C_{Max} == 0$
        break for
    else
        exchange the placements of $d_{Max1}$ and $d_{Max2}$
return $P$
```

(c)

```
Procedure SAMPLE
Input: original broadcast program $P$, parameters $x$, $y$
Output: optimized broadcast program $P'$
for y_count = 1 to $y$
    $\Delta C_{Max} = 0$, $d_{Max1}$=null, $d_{Max2}$=null, x_count=0
    Create a set $S$ storing all possible exchanging pairs
    while $|S| > 0$
        Randomly select a pair $(i_1, i_2)$ from $S$
        Remove $(i_1, i_2)$ from $S$
        if $p_{i1} \neq p_{i2}$
            calculate $\Delta C(i_1,i_2)$
            if $\Delta C_{Max} < \Delta C(i_1,i_2)$
                $\Delta C_{Max} = \Delta C(i_1,i_2)$, x_count ++
                $d_{Max1}= d_{i1}$, $d_{Max2}= d_{i2}$
        if x_count == $x$
            break while
    if $\Delta C_{Max} == 0$
        break for
    else
        exchange the placements of $d_{Max1}$ and $d_{Max2}$
return $P$
```

(d)

Fig. 6. The proposed optimization procedures in sequential data broadcasting. (a) Algorithmic form of procedure BASIC. (b) Algorithmic form of procedure SAMPLE. (c) Algorithmic form of procedure ITERATION. (d) Algorithmic form of procedure SCI.

we reduce the complexity by reformulating $\Delta C(u, v)$. As shown in (7), the nested summation is decomposed into exactly *four* simple summations. Therefore, it only requires $O(N)$ to derive the cost reduction after the reformulation.

According to the cost reduction of exchanging two items in different placements, we first design procedure BASIC to optimize the broadcast program generated by algorithm OLS. The algorithmic form of procedure BASIC is outlined in Fig. 6a. Procedure BASIC will be executed iteratively. In each iteration, it will reduce the cost as much as possible. More specifically, procedure BASIC aims to find out the best exchanging operation that achieves the maximum cost reduction. To achieve the maximum cost reduction, procedure BASIC will search the exchanging operations of *any* two items. If the two items i_1 and i_2 are located in different placements, the cost reduction $\Delta C(i_1, i_2)$ will be calculated according to (7). Notice that algorithm BASIC also allows more than two items to exchange their positions. However, calculating the cost reduction of exchanging more than two items requires much higher complexity. It is thus inefficient and impractical for procedure BASIC to exchange more than two items. At the end of the iteration, the exchanging operation with the maximum cost reduction will be chosen to update the broadcast program. In the next iteration, the updated broadcast program will be considered. Procedure BASIC continues until the *local optimal* state is reached, where no exchanging operation can result in the cost reduction. In each iteration, the searching complexity is $O(N^2)$, whereas the complexity of calculating the cost reduction is $O(N)$. It requires $O(N^3)$ complexity to determine the exchanging operation with the maximum cost reduction.

In order to enhance the flexibility of procedure BASIC, we design two extended procedures from BASIC. The first procedure, SAMPLE, is designed to achieve the local optimum more efficiently with a control parameter x. The algorithmic form of procedure SAMPLE is outlined in Fig. 6b. Unlike procedure BASIC, which examines all possible exchanging operations, procedure SAMPLE only examines part of them. In each iteration, procedure SAMPLE first creates a set S, which stores all possible exchanging pairs. Next, procedure SAMPLE will randomly select x *samples* capable of reducing the cost. At the end of the iteration, the exchanging pair that induces the maximum cost reduction will be chosen to update the broadcast program. In the next iteration, procedure SAMPLE will be executed according to the updated broadcast program. Like procedure BASIC, procedure SAMPLE terminates its execution once no exchanging operation can reduce the cost. Although procedure SAMPLE can also return a *local optimal* solution, different values of x will result in different qualities of the broadcast program. When the parameter x is large, the SAMPLE procedure will perform similarly to the BASIC procedure at the end of the iteration. The advantage of the SAMPLE procedure over the BASIC procedure will be more distinguishable at a smaller x.

It is noted that the sampling techniques are widely used in efficiently processing large numbers of data items. For example, in clustering analysis, the Clustering Large Applications (CLARA) method [18] can be viewed as the sampling-based algorithm. Like CLARA, the number of selected candidates is much smaller than the size of the search space, which is helpful in reducing the execution time. On the other hand, the major difference between our SAMPLE procedure and CLARA is that given the parameter x, CLARA will examine exactly x samples. However, SAMPLE may examine more than x samples. The reason is that SAMPLE selects x samples that contribute to the reduction of the access time. Since not all of the candidates lead to the reduction, some of the examined samples may not be selected. Due to this feature, although it takes more execution time in selecting x effective samples, it can be guaranteed that the SAMPLE algorithm stops its execution only when no exchanging operation can result in cost reduction.

Another extended procedure is named ITERATION. With parameter y, which determines the number of iterations, procedure ITERATION can compromise between effectiveness and efficiency. The algorithmic form is outlined in Fig. 6c. Although BASIC and SAMPLE can reach the local optimum, there is no guarantee of execution time since the number of iterations cannot be estimated. To remedy this unpredictably, procedure ITERATION is allowed to be iterated at most y times. Once the procedure iterates itself for y times, it is forced to be terminated. The most advantageous feature of ITERATION is that the optimization complexity is $O(N^3)$, which enables ITERATION to be executed in polynomial time. Moreover, ITERATION also performs well in approximating the quality of procedure BASIC. The reason is as follows: Although it requires hundreds or even thousands of iterations for BASIC to reach the local optimum, the preceding iterations provide a much higher marginal effect on reducing the cost than succeeding ones.

Since the two extended procedures, SAMPLE and ITERATION, enhance the efficiency of BASIC in different aspects, we can further design another optimization procedure by integrating the advantageous features of SAMPLE and ITERATION. The proposed procedure is named SCI. The algorithmic form is outlined in Fig. 6d. In SCI with control parameters (x, y), the parameter x is to determine the number of samples in each iteration, whereas the parameter y is to limit the number of iterations. It is clear that SCI provides more flexibility compared to SAMPLE and ITERATION. More specifically, three extreme cases, $\mathrm{SCI}(x, \infty)$, $\mathrm{SCI}(\infty, y)$, and $\mathrm{SCI}(\infty, \infty)$, are exactly identical to SAMPLE, ITERATION, and BASIC, respectively. Therefore, we suggest that procedure SCI should be employed for optimization in the MULS framework to provide multilevel service quality in the sequential data broadcasting environment.

3.4 Implementation Issues of the MULS Framework

The proposed MULS framework is based on a simplified model in the mobile computing environment. In practice, the query profile and the broadcast database may be more complicated. With several extensions employed, our work can also be suitable for some complicated environments. In the following, we will discuss several practical issues of the MULS framework.

3.4.1 Consistency Maintenance

In the sequential data broadcasting environment, the contents of the items will be updated unpredictably. When the content of one item is updated, those clients who keep the item before the session is completed should download the item again to avoid the inconsistency. In order to support the consistency maintenance, several extensions should be made on the MULS framework. First, the invalidation message [3] should be broadcast so as to notify the clients that data items have been updated. Since the invalidation message is of small size, it can be disseminated between the time slots of broadcasting two consecutive items without affecting the service quality.

When a user receives the invalidation message, he or she will check the current state of receiving items. If some of the received items are invalid, the user will stop the receiving session. After that, the original query will be refined by the client and resent to the server. Note that the objective of generating a refined query is to make sure that all of the invalid items should be downloaded again before the residual items are downloaded. The residual items stand for the items that appear in the original query but were not downloaded due to the breach of the receiving procedure. Specifically, the original query q_i will be refined as $q_i^{(I)} + q_i^{(R)}$, where $q_i^{(I)}$ represents the query for updated items in the original order, and $q_i^{(R)}$ denotes the query for residual items. Finally, the new broadcast program generated by the server will be based on the refined query submitted by the clients. For example, let the original query be $q_i = \{d_1 -> d_2 -> d_3 -> d_4 -> d_5\}$. When the user detects the invalidation of d_1 and d_3 after downloading $\{d_1, d_2, d_3\}$, the current downloading session will be stopped, and a new query $q_i' = \{d_1 -> d_3 -> d_4 -> d_5\}$ will be issued. Another alternative is a lazy approach by which the clients just discard the invalidated data items.

3.4.2 Dealing with Repeated Data in a Query

Although the design of the MULS framework is based on the assumption that each item will appear at most once in one query, in practice, the items may be repeated in a query. It is noted that for repeated items in a query, the users will use the local cache to answer the probing to the same item within the same query. Therefore, in the design of the scheduling algorithms, we should take the effect of repeated items into account. Under the circumstance that one item will appear more than once, the server will refine the query by keeping the first item and removing the repeating items. For example, consider a query $q_i = \{d_2 -> d_1 -> d_3 -> d_1 -> d_2\}$. When the server receives q_i, this query will be refined as $q_i' = \{d_2 -> d_1 -> d_3\}$. The broadcast program will be generated according to q_i' instead of q_i. Also, notice that once one client starts to retrieve the items in q_i, d_2 and d_1 will be kept in the cache at the client side after their first retrieval. After d_3 is retrieved from the broadcast database, the cache will return d_1 and d_2 so as to satisfy the access of the sequential query.

3.4.3 Broadcasting Heterogeneous Items

In the proposed MULS framework, our analytical model and scheduling algorithms are based on the assumptions that the items are of the same size. In practice, the size of one item may be different from the other. When the items are of different sizes (that is, heterogeneous data [4] are broadcast), a practical solution is to divide each item into several pieces in such a way that each piece is of the same size and employ the proposed scheduling algorithm in generating the broadcast program. Due to the sequential property of retrieving the data, all the pieces of the same item will be downloaded consecutively. Since the client will complete the retrieval of all pieces of the current item before the first piece of the next item is accessed, only a small cache at the client side is required.

4 Experimental Evaluation

To inspect the performance of the MULS framework, several experiments are conducted. We evaluate the issue of effectiveness and efficiency by varying the number of items, the number of channels, and the control parameters (x, y). In the beginning, the analytical model in this paper will be validated using the event-driven simulation. We will describe the simulation environment in Section 4.2. In

TABLE 3
Values of the Simulation Parameters

Parameters	Default Values
Number of the items (N)	240
Number of the queries ($\|Q\|$)	100
Number of the channels (K)	12
Average length of a query (l)	10
Skewness parameter (θ)	1.0
Bandwidth of each broadcast channel (B)	80K bytes/sec
Size of each data item (s)	8K bytes

Section 4.3, the effectiveness issue will be analyzed with the major parameters varied. In Section 4.4, the efficiency study will be presented. In Section 4.5, we will discuss the effect of parameters x and y of procedure SCI. In Section 4.6, the adaptiveness of the SCI algorithm will be analyzed. Finally, remarks will be made in Section 4.7.

4.1 Simulation Environment

Table 3 summarizes the definitions for primary simulation parameters. To reflect the access skew in the database system, the access probabilities of queries are generated by the Zipf distribution [30] $f_i = (\frac{1}{i})^{\theta} / \sum_{j=1}^{N} (\frac{1}{j})^{\theta}$, where θ is a *skewness parameter*, and $1 \leq i \leq N$. Note that a large θ indicates highly skewed access patterns in the mobile computing environment. When $\theta = 0$, the access probabilities of the queries are uniformly distributed. By default, the value of θ is set to be 1 since it is observed in [24] that θ is usually larger than 1 for busy Web sites. The default value of N is based on the number of Web pages stored in a regular server. We also set the default value of the average query length l to be 10. The reason is that if l is smaller than $\lceil \frac{N}{K} \rceil$, most mobile users can retrieve all the items of interest in one broadcast cycle with a well-designed scheduling algorithm employed. The query profile is generated based on the approach mentioned in [23], in which the interrelationship among the items can be modeled as a dependency graph. From the experimental results, we observe that the number of the queries is viewed as a minor factor affecting the performance. Therefore, in the following experiments, the number of queries is fixed to be 100. In addition to generating the query profile synthetically, we also use real data sets to validate the proposed approaches. The real data sets, regarded as *alarm queries*, are issued by the base stations and collected by a major telecommunication company.

We implement the relevant approaches using Java language[9] and execute the programs on an IBM compatible PC with a Pentium IV 3.2-GHz processor and 1 Gbyte of RAM. The statistics have been collected by averaging the values in 10 experiments with different query profiles. During the experiments, we inspect the performance of the following approaches:

1. OLS, indicating that the broadcast program is generated by algorithm OLS without optimization,
2. BASIC, indicating that the broadcast program is generated by OLS and optimized by BASIC,
3. SAMPLE, in which the optimization procedure SAMPLE is employed with parameter $x = 1$ (denoted by SAM),
4. ITERATION, in which ITERATION is employed with parameter $y = 20$ (denoted by ITER), and
5. SCI, in which SCI is employed with parameters $x = 1$ and $y = 20$.

For comparison purposes, we also implement GA [13], MQEM [19], the RTS method [20], and the greedy algorithm (denoted by GRE) proposed by Yee et. al. [28]. Since the design of GRE is based on the assumption that each user query contains only one item, we divided the behavior of accessing the sequential data into a sequence of sessions. In each session, only one item will be accessed.

4.2 Accuracy of Analytical Model

In the first experiment, we validate our analytical model used in Section 2.2. Although $T_A(Q)$ derived in (2) enables us to evaluate the quality of a broadcast program, however, this analytical model only provides an approximation of the average access time. To validate the approximation accuracy, we implement an event-driven simulation. We consider that the broadcast program is updated every minute. The current query profile is generated by collecting the statistics of the queries using a sliding window with 1-minute window size. We discuss the effect of three parameters: the arrival rates of the queries, the size of query profile ($|Q|$), and the skewness parameter θ. The approximation error is defined as $Err = (|T_A^{(Approx)} - T_A^{(Real)}|)/T_A^{(Real)}$, where $T_A^{(Approx)}$ denotes the average access time obtained by (2), and $T_A^{(Real)}$ represents the real average access time measured in the experiment. As mentioned earlier, the query profile is generated according to the statistics in the last minute. Since the current access pattern may not exactly be the same as that in the query profile, there will be some approximation errors using our analytical model. On the other hand, in the real world, each mobile user may start to listen to the broadcast channel at different time points, which also influences his or her access time. As shown in Fig. 7, our analytical model can mostly lead to less than 1 percent approximation error. Moreover, under some special cases such as high arrival rates of queries, highly skewed access patterns, and large numbers of queries, the analytical model in (2) will have outstanding approximation accuracy. From the experimental results, we can simply use (2) to evaluate the quality of a broadcast program instead of measuring the waiting time of each mobile user.

9. In the experimental study, we focus on comparing the relative performances among different algorithms. Therefore, using the Java language will not affect the relative merit of the methods evaluated.

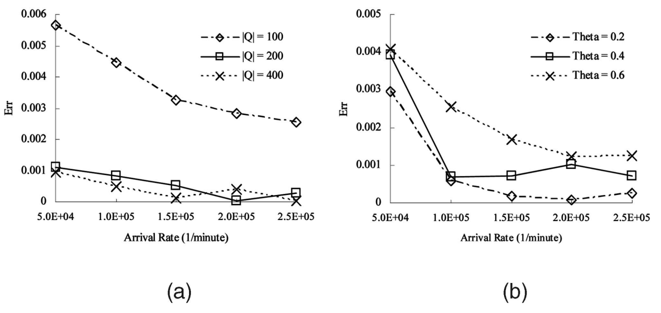

Fig. 7. Error of the analytical model. (a) Error of analytical model at different sizes of query profile. (b) Error of analytical model at different skewness parameters.

4.3 Effectiveness Analysis

Next, we discuss the effectiveness issue with major parameters varied. We discuss the effect of the number of broadcast items (N), the number of broadcast channels (K), the skewness parameter (θ), and the average length of each query (l). The amount of $T_A(Q)$ is used to assess the quality of broadcast programs generated by different algorithms.[10] In Fig. 8a, we observe that the access time goes up as the number of items increases. This phenomenon can be explained as follows: When more items need to be broadcast, each channel requires a longer broadcast cycle to deliver the items. The increase of broadcast cycle will raise the access time of mobile users. In Fig. 8b, at a large value of θ, since the access distribution is highly skewed, the generated broadcast program will tend to satisfy most of the clients, which is also helpful to reduce the average access time. As for Figs. 8c and 8d, we show the effect of parameter K on both synthetic and real data sets. The clients will spend less time to download all the items of interest when more channels are available, which agrees with our intuition since the increase of the network bandwidth causes the access time to decrease.

Throughout Fig. 8, it can be observed that among all optimization procedures, BASIC can in general achieve the lowest access time. On the other hand, although SAMPLE(1) only selects one sample in each iteration, the quality is still very close to that of BASIC. It is because SAMPLE(1) also iterates itself until the local optimum is reached. As for ITERATION(20), we can observe that the preceding 20 iterations contribute over 50 percent of improvement from OLS to BASIC. During the experiments, there are approximately 350 iterations before BASIC reaches the local optimum. This phenomenon shows that the marginal effect of the preceding iterations is definitely more significant than that of the succeeding iterations. The integration of SAMPLE(1) and ITERATION(20), that is, SCI(1, 20), achieves inferior quality compared to SAMPLE(1) and ITERATION(20). The reason is that unlike ITERATION(20), which searches all possible moving operations, SCI(1, 20) only selects *one* sample capable of reducing the cost. The marginal effect of the preceding iterations is thus insignificant.

It is noted that algorithm OLS indicates the worst case of the average access time since no optimization procedures are performed. Compared to other competitive approaches, algorithm OLS still achieves outstanding performances. Due to the ignorance of the *ordered dependency* among broadcast items, algorithm MQEM results in the poorest performance. As for GA, since the results of the stochastic search may be bound by a false optimal solution, the quality of the broadcast program cannot be guaranteed. On the other hand, since algorithm GRE is designed for the situations in which only one item is contained in each query, the performance of GRE will get worse as the value of l increases. Among all competitive methods, algorithm RTS achieves the most acceptable solutions since the characteristics of sequential relationship is explored via a directed graph. However, the performance of RTS is still inferior to that of algorithm OLS.

4.4 Efficiency Analysis

In this experiment, we inspect the efficiency of relevant approaches. We report the execution time of the server on executing the relevant algorithms. We use millisecond as the unit of execution time. Fig. 9a indicates that as the number of broadcast items grows, the execution time of the simulators goes up. It is effortless to explain this phenomenon since the increase of N will enlarge the search space for either a deterministic search such as the proposed optimization procedures or a stochastic search like GA. With the performance of GA as the yardstick, we can observe that procedure SCI(1, 20) and algorithm MQEM are much more efficient, whereas procedure BASIC takes more execution time. As for procedure ITERATION(20), the performance is very close to GA. As shown in Fig. 9b, it is observed that the execution time of BASIC and SAMPLE(1) decreases as K increases. The reason is that the number of iterations for BASIC and SAMPLE(1) to reach the local optimum is larger when K is small. Also, the decrease of K will raise the length of the broadcast cycle L. Since at a larger L, there will be a larger number of choices to put the items, it will thus require more iterations to determine the suitable placements for items to be allocated.

4.5 Effect of Control Parameters

We discuss the effect of the parameters x and y empirically. We only inspect procedure SCI since the other optimization procedures are regarded as special cases of SCI. During the experiment, x is varied from 1 to 50, whereas y is varied from 0 to 80. Fig. 10a depicts the improvement of the access time as the increase of x and y. We can observe that the curve SCI(1, y) goes down as y increases. Compared to SCI(1, y), the curve SCI(5, y) has a much significant improvement when y is smaller (for example, $y = 10$). However, when y is larger, the improvement saturates. This phenomenon illustrates that the marginal effect of the preceding iterations is more significant than that of succeeding iterations. According to SCI(10, y) and SCI(50, y), we can also observe that the increase of x is not always helpful to result in a distinguishable improvement due to the saturation of the performance. Fig. 10b shows the execution time in the

10. In this paper, we do not rely on the specific assumptions of mobile devices. Instead, our analytical models and algorithms can be suitable for the devices for general purposes. Therefore, in our experiments, instead of comparing the data processing scheme in different devices, we focus on observing the average access time for users before the data items are downloaded.

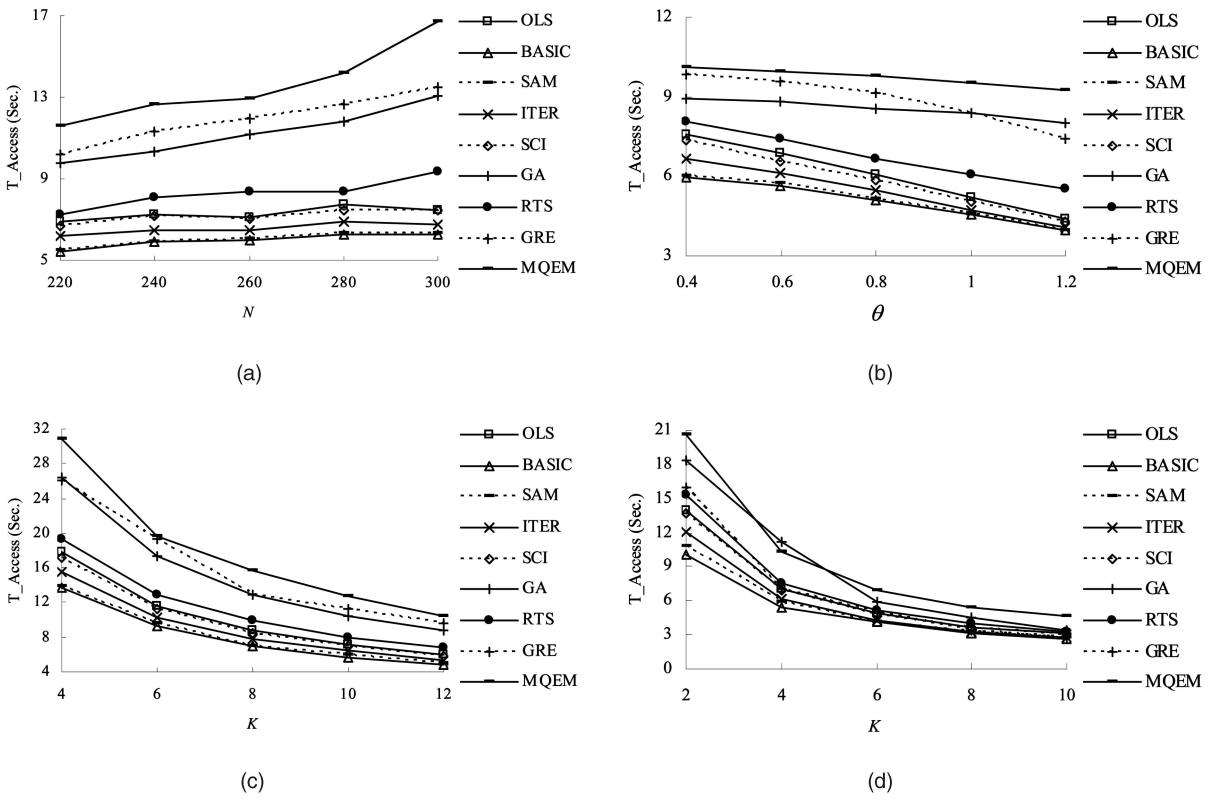

Fig. 8. Effectiveness of the MULS framework. (a) The average access time with N varied (synthetic data set). (b) The average access time with θ varied (synthetic data set). (c) The average access time with K varied (synthetic data set). (d) The average access time with K varied (real data set).

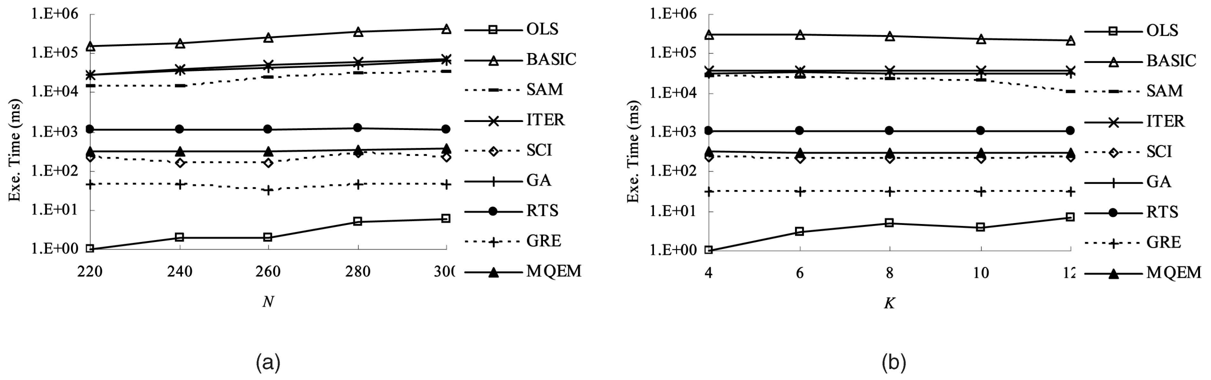

Fig. 9. The efficiency analysis of relevant algorithms. (a) Execution time with N varied. (b) Execution time with K varied.

exponential scale as different x and y are set in procedure SCI. It is noted that the increase of y will raise the execution time linearly, whereas the increase of x will raise the execution time drastically. This phenomenon can be explained by taking SCI(1, y) and SCI(5, y), for example. Since SCI(5, y) has better quality than SCI(1, y), it will take much more time for SCI(5, y) to find out five samples capable of reducing the access time because many exchanging operations cannot contribute to the improvement.

4.6 Adaptiveness of the MULS Framework

In the final experiment, we discuss the adaptiveness of the MULS framework in both dynamic and static environments. During the experiment, we vary the skewness parameter θ at different time points so as to reflect the change of the access patterns. We inspect the performances of four parameter settings: OLS (that is, SCI(0, 0)), SCI(10, 20), SCI(50, 50), and BASIC (that is, SCI(∞, ∞)). In addition, two adjusting approaches, OFFLINE and NoAdjust, are also implemented for comparison purposes. The OFFLINE approach will generate the broadcast program in the offline manner, whereas the NoAdjust approach will always adopt the original schedule no matter how the access patterns are varied. Note that OFFLINE indicates the lower bound for each approach to generate the broadcast program in response to the dynamics of access patterns. Fig. 11 (respectively, Fig. 12) shows the variance of θ and the corresponding performances in the static (respectively, dynamic) environment. From the experimental results, it can be seen that in the static environment in which the access patterns change tardily, the MULS framework with larger parameters is capable of generating the broadcast programs

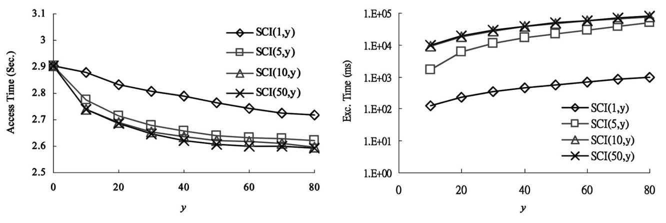

Fig. 10. The access time and the execution time of $\mathrm{SCI}(x, y)$ with different control parameters. (a) Average access time of SCI with parameters x and y varied. (b) Execution time of SCI with parameters x and y varied.

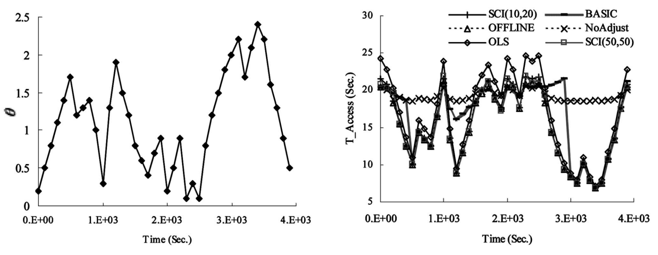

Fig. 12. The adaptiveness of the MULS framework in the dynamic environment. (a) Variance of θ over time. (b) Performance of SCI(10, 20) and BASIC.

of high quality. On the other hand, in the dynamic environment in which the access patterns change rapidly, since it takes a high execution time for BASIC, the broadcast program would be updated infrequently, which will affect the service quality for users to access data. On the other hand, the MULS framework with small parameters can respond well and achieve high adaptivity.

4.7 Remarks

From the above experiments, we observe that the broadcast program generated by OLS can result in better quality than GA, MQEM, RTS, and GRE. Moreover, algorithm OLS costs only $O(|Q| \log |Q|) + O(N \log N)$ to schedule the items in sequential data broadcasting. These advantageous features make algorithm OLS very suitable for online data broadcasting. In order to optimize the broadcast program generated by algorithm OLS, it is suggested that procedure SCI should be employed in our MULS framework. The reasons are as follows: First, procedure SCI is more flexible than others. By tuning the parameters x and y, SCI can achieve the same quality as BASIC, SAMPLE, and ITERATION. Second, SCI can provide much more different levels of service quality than other optimization procedures. Therefore, SCI can more easily strike a compromise between efficiency and effectiveness requirements for information systems. Moreover, from the efficiency analysis in Section 4.2, SCI is more scalable so that it can still perform well for broadcasting a large-scale database. In the dynamic environment in which the access patterns and information contents change rapidly, the broadcast program should be regenerated frequently. Under this circumstance, the parameters x and y can be tuned small so as to generate an online broadcast program. As for the static environment in which the query profile and the database are updated infrequently, an optimized broadcast program can remain the same for a long while. We can thus adopt larger values of x and y to generate an offline broadcast program with enhanced quality.

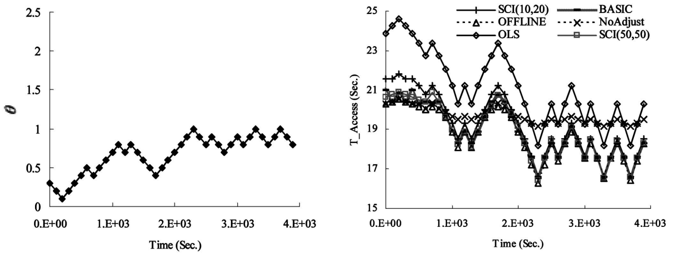

Fig. 11. The adaptiveness of the MULS framework in the static environment. (a) Variance of θ over time. (b) Performance of SCI(10, 20) and BASIC.

5 Conclusion

In this paper, we focus on generating broadcast programs in the *sequential data broadcasting* environment. To provide multilevel service quality for different effectiveness and efficiency requirements, we propose a general framework named MULS, which contains algorithm OLS and an optimization procedure, SCI. According to the experimental results, algorithm OLS can generate broadcast programs with satisfactory quality. As for the optimization procedure, SCI performs well in its scalability and is very suitable to broadcast items within a larger database. By tuning the parameters x and y, the proposed framework is able to generate broadcast programs with multilevel service quality for different effectiveness and efficiency requirements in static and dynamic environments. In our future research, we will focus on two important issues. The first one is the real-time data broadcasting in which the users may specify that the items of interest should be downloaded within a specific time constraint. The second one is the skewed data broadcasting. As mentioned Section 2, although the skewed data broadcasting will achieve better performances than the flat broadcasting, due to the unpredictable average access time, we do not consider the skewed broadcasting in this paper. We plan to develop the scheduling approaches via skewed broadcasting for some specialized broadcast profile.

Acknowledgments

This work was supported in part by the National Science Council of Taiwan, ROC, under Contracts NSC93-2752-E-002-006-PAE.

References

[1] Sony Corp., http://www.sony.co.jp/, 2006.
[2] S. Acharya, R. Alonso, M.J. Franklin, and S.B. Zdonik, "Broadcast Disks: Data Management for Asymmetric Communications Environments," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '95)*, pp. 199-210, May 1995.

[3] S. Acharya, M.J. Franklin, and S.B. Zdonik, "Disseminating Updates on Broadcast Disks," *Proc. 22nd Int'l Conf. Very Large Data Bases (VLDB '96)*, pp. 354-365, Sept. 1996.
[4] S. Acharya and S. Muthukrishnan, "Scheduling On-Demand Broadcasts: New Metrics and Algorithms," *Proc. MobiCom '98*, pp. 43-54, 1998.
[5] D. Aksoy and M. Franklin, "RxW: A Scheduling Approach for Large-Scale On-Demand Data Broadcast," *IEEE/ACM Trans. Networking*, vol. 7, no. 6, pp. 846-860, 1999.
[6] D. Aksoy, M.J. Franklin, and S. Zdonik, "Data Staging for On-Demand Broadcast," *Proc. 27th Int'l Conf. Very Large Data Bases (VLDB '01)*, pp. 571-580, Sept. 2001.
[7] D. Aksoy and M.S. Leung, "Pull vs. Push: A Quantitative Comparison for Data Broadcast," *Proc. IEEE Global Telecomm. Conf. (GLOBECOM '04)*, 2004.
[8] M.-S. Chen, P.S. Yu, and K.-L. Wu, "Optimizing Index Allocation for Sequential Data Broadcasting in Wireless Mobile Computing," *IEEE Trans. Knowledge and Data Eng.*, vol. 15, no. 1, Jan./Feb. 2003.
[9] Y. Chung and M. Kim, "Efficient Data Placement for Wireless Broadcast," *Distributed and Parallel Database*, vol. 9, no. 2, Mar. 2001.
[10] Y.D. Chung and M.H. Kim, "QEM: A Scheduling Method for Wireless Broadcast Data," *Proc. Sixth Int'l Conf. Database Systems for Advanced Applications (DASFAA '99)*, 1999.
[11] C.-H. Hsu, G. Lee, and A.L.P. Chen, "A Near Optimal Algorithm for Generating Broadcast Programs on Multiple Channels," *Proc. 10th ACM Int'l Conf. Information and Knowledge Management (CIKM '01)*, pp. 303-309, Nov. 2001.
[12] C.L. Hu and M.S. Chen, "On-Line Scheduling Sequential Objects for Dynamic Information Dissemination," *Proc. IEEE Global Telecomm. Conf. (GLOBECOM '05)*, 2005.
[13] J.-L. Huang and M.-S. Chen, "Broadcasting Dependent Data for Ordered Queries without Replication in a Multi-Channel Mobile Environment," *Proc. 19th IEEE Int'l Conf. Data Eng. (ICDE '03)*, Mar. 2003.
[14] H.P. Hung and M.S. Chen, "On Exploring Channel Allocation in the Diverse Data Broadcasting Environment," *Proc. 25th IEEE Int'l Conf. Distributed Computing Systems (ICDCS '05)*, 2005.
[15] H.P. Hung, J.W. Huang, J.L. Huang, and M.S. Chen, "Scheduling Dependent Items in Data Broadcasting Environments," *Proc. 21st Ann. ACM Symp. Applied Computing (SAC '06)*, 2006.
[16] T. Imielinski, S. Viswanathan, and B.R. Badrinath, "Data on Air: Organization and Access," *IEEE Trans. Knowledge and Data Eng.*, vol. 9, no. 3, pp. 353-372, May/June 1997.
[17] J.-L. Huang and M.-S. Chen, "Dependent Data Broadcasting for Unordered Queries in a Multiple Channel Mobile Environment," *IEEE Trans. Knowledge and Data Eng.*, vol. 16, no. 6, June 2004.
[18] L. Kaufman and P.J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley & Sons, 1990.
[19] G. Lee and S.C. Lo, "Broadcast Data Allocation for Efficient Access of Multiple Data Items in Mobile Environments," *Mobile Networks and Applications*, vol. 8, no. 4, 2003.
[20] C.M. Liu and K.F. Lin, "Efficient Scheduling Algorithms for Disseminating Dependent Data in Wireless Mobile Environments," *Proc. IEEE Int'l Conf. Wireless Networks, Comm., and Mobile Computing (WIRELESSCOM '05)*, 2005.
[21] S.-C. Lo and A.L. Chen, "Optimal Index and Data Allocation in Multiple Broadcast Channels," *Proc. 16th IEEE Int'l Conf. Data Eng. (ICDE '00)*, pp. 293-302, 2000.
[22] F. Martinez, J. Gonzalez, and I. Stojmenovic, "A Parallel Hill Climbing Algorithm for Pushing Dependent Data in Clients-Providers-Servers Systems," *Proc. Seventh Int'l Symp. Computer and Comm. (ISCC '02)*, 2002.
[23] A. Nanopoulos, D. Katsaros, and Y. Manolopoulos, "Effective Prediction of Web-User Accesses: A Data Mining Approach," *Proc. 12th ACM SIGKDD Workshop Web Mining and Web Usage Analysis (WebKDD '01)*, 2001.
[24] V. Padmanabhan and L. Qiu, "The Content and Access Dynamics of a Busy Web Site: Findings and Implications," *Proc. ACM Conf. Applications, Technologies, Architectures, and Protocols for Computer Comm. (SIGCOMM '00)*, 2000.
[25] W.-C. Peng and M.-S. Chen, "Efficient Channel Allocation Tree Generation for Data Broadcasting in a Mobile Computing Environment," *Wireless Networks*, vol. 9, no. 2, pp. 117-129, 2003.
[26] N. Prabhu and V. Kumar, "Data Scheduling for Multi-Item and Transactional Requests in On-Demand Broadcast," *Proc. Sixth Int'l Conf. Mobile Data Management (MDM '05)*, 2005.
[27] A. Si and H.V. Leong, "Query Optimization for Broadcast Database," *Data and Knowledge Eng.*, vol. 23, no. 9, 1999.
[28] W.-G. Yee, S.B. Navathe, E. Omiecinski, and C. Jermaine, "Efficient Data Allocation over Multiple Channels at Broadcast Servers," *IEEE Trans. Computers*, vol. 51, no. 10, pp. 1231-1236, Oct. 2002.
[29] B. Zheng, X. Wu, X. Jin, and D.L. Lee, "TOSA: A Near-Optimal Scheduling Algorithm for Multi-Channel Data Broadcast," *Proc. Sixth Int'l Conf. Mobile Data Management (MDM '05)*, 2005.
[30] G.K. Zipf, *Human Behaviour and the Principle of Least Effort.* Addison-Wesley, 1949.

Hao-Ping Hung received the BS degree from the Department of Electrical Engineering, National Taiwan University, Taipei, in 2001, and the PhD degree from the Graduate Institute of Communication Engineering, National Taiwan University, in January 2007. Now, he serves as a senior engineer at CyberLink Corp. His research interests include mobile computing, resource allocation in the wireless environment, multimedia networking, and data streams.

Ming-Syan Chen received the BS degree in electrical engineering from the National Taiwan University, Taipei, and the MS and PhD degrees in computer, information, and control engineering from the University of Michigan, Ann Arbor, in 1985 and 1988, respectively. He is now a distinguished professor jointly appointed by the Department of Electrical Engineering, the Department of Computer Science and Information Engineering Department, and also the Graduate Institute of Communication Engineering, National Taiwan University. He was a research staff member at IBM T.J. Watson Research Center, Yorktown Heights, New York, from 1988 to 1996. He has published more than 240 papers in his research areas. In addition to serving as program chairs/vice-chairs and keynote/tutorial speakers in many international conferences, he was an associate editor of the *IEEE Transactions on Knowledge and Data Engineering* and the *Journal of Information Science and Engineering*, is currently on the editorial board of the *Very Large Data Base Journal*, the *Knowledge and Information Systems Journal*, and the *International Journal of Electrical Engineering*), and is a distinguished visitor of the IEEE Computer Society for Asia-Pacific from 1998 to 2000 and 2005 to 2007. He holds or has applied for 18 US patents and seven ROC patents in his research areas. His research interests include database systems, data mining, mobile computing systems, and multimedia networking. He is a recipient of the National Science Council (NSC) Distinguished Research Award, Pan Wen Yuan Distinguished Research Award, Teco Award, Honorary Medal of Information, and K.-T. Li Research Breakthrough Award for his research work and the Outstanding Innovation Award from IBM Corporate for his contribution to a major database product. He also received numerous awards for his research, teaching, inventions, and patent applications. He is a fellow of the ACM and the IEEE.