
Distributed and Parallel Databases, 9, 133–150, 2001

Effective Data Placement for Wireless Broadcast

YON DOHN CHUNG ydchung@dbserver.kaist.ac.kr
MYOUNG HO KIM mhkim@dbserver.kaist.ac.kr
Division of Computer Science, Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology, 373-1, Kusung-dong, Yusung-gu, Taejon, 305-701, Korea

Received September 30, 1998; Revised January 27, 1999; Accepted December 22, 1999

Recommended by: Jin Jing

Abstract. This paper investigates how to place data objects on air for wireless broadcast such that mobile clients can access the data in short latency. We first define and analyze the problem of wireless data placement, and also propose a measure, named *Query Distance* (*QD*), which represents the coherence degree of data set accessed by a query. We show that the problem is *NP*-complete, and then propose an effective data placement method that constructs the broadcast schedule by appending each query's data set in greedy way. We show through performance experiments that the proposed method reduces the access time of mobile query.

Keywords: data placement, data clustering, multi-point query, data broadcasting, mobile computing

1. Introduction

There are two important parameters related with wireless data broadcasting. They are access time and tuning time. The access time is the amount of time elapsed from the moment a client submits a query to the receipt of the data of his (her) interest on the broadcast channel. The tuning time is the amount of time spent by a client listening to the channel. As the tuning time for accessing data is determined by the amount of time spent being in active mode (plus a small amount for being in doze mode), this will determine the power consumed by the client to retrieve the required data.

There have been some researches on reducing access time such as caching and nonuniform broadcasting [1, 2, 9]. Researches on reducing tuning time such as indexing and hashing have also been made in the past [4, 7, 8]. In this work we mainly focus on effective data placement for wireless broadcast such that the access time of the queries issued by mobile clients can be reduced.

In wireless systems the server communicates the mobile clients with one-way (broadcasting) and two-way (request and reply) method. Because of the inherent restrictions of wireless communication, such as bandwidth and energy restrictions, the broadcasting method is preferred [7]. Therefore, the server analyzes the data access patterns of the clients' requests and broadcasts the data objects of popularity. In wireless data broadcasting the server repeatedly, in a given period, broadcasts data stream commonly called *bcast* to unspecified clients and the client accesses the data of interest in the broadcast stream.

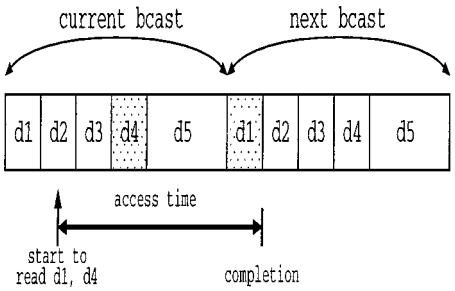

Figure 1. A query onto the wireless broadcast.

The way a mobile client accesses the broadcast stream is illustrated in figure 1, where the server broadcasts a set of data objects $\{d1, d2, d3, d4, d5\}$ in one *bcast*. Note that whenever broadcasting of the current *bcast* ends, the next *bcast* is broadcasted continuously. Suppose a client issues a query retrieving $d1$ and $d4$.[1] In the figure the client has to wait for the next *bcast* to access $d1$ because the data object $d1$ is passed over at the time when the client's query is issued. However, if the data in the *bcast* is placed as $\langle d2, d3, d1, d4, d5\rangle$, then the client can retrieve $d1$ and $d4$ in the current *bcast*. This implies that how to place data in the *bcast* is important for access time reduction to mobile clients.

The works [1], [2], [10] and [9] reduce clients' access time by efficient organization of the broadcast data stream. *Broadcast Disks* [1, 2] approach analyzes the access preference of each data object and differentiates the delivery frequencies of data objects. That is, the more popular data objects are more frequently broadcasted. This way of broadcasting increases the *bcast* length, so it gives longer average access time to the clients who access less popular data objects. The approach does not consider that a query accesses more than one data object like in figure 1.

The scheduling method in [9] makes the broadcast schedule by using stochastic model. It considers the access frequencies of data objects and controls their delivery intervals. However, the approach does not consider the relationship between data objects when a query accesses more than one data object. Also this approach has similar shortcomings of *Broadcast Disks* approach in the above by increasing the broadcast cycle.

This paper investigates the placement of wireless data for uniform broadcasting environment i.e., the data objects are not replicated in a single *bcast*. After defining *Query Distance* measure, which represents the coherence degree of data objects that a query accesses, we propose a data placement method. The method constructs the broadcast schedule by appending the data objects that each query accesses. In constructing the schedule, we minimize the *Query Distance* of each query in greedy manner.

The rest of the paper is organized as follows. In Section 2 we define data placement problem for wireless broadcast and propose a new measure for wireless data placement. We propose a scheduling method with some illustrative examples in Section 3, and evaluate the performance of the proposed method in Section 4. We conclude the paper with some directions for further research in Section 5.

Notation	Meaning
d_i	a data object to be broadcasted
$\|d_i\|$	the size of d_i; the number of packets
$\mathcal{D}$	the set of data objects d_i; $\{d_1, d_2, \cdots, d_N\}$
B	the size of a broadcast stream i.e., $\sum \|d_i\|$, $\forall d_i \in \mathcal{D}$
q_i	a query that is issued on the broadcast data stream
$QDS(q_i)$	the set of data objects that q_i accesses
$freq(q_i)$	the reference frequency of q_i
$\mathcal{Q}$	the set of queries q_i; $\{q_1, q_2, \cdots, q_M\}$
σ	the broadcast schedule of $\mathcal{D}$
	i.e., the sequence of data broadcasting.
	σ is denoted by $< d_i, d_j, \cdots, d_k >$.

Figure 2. Symbol definitions.

2. Problem definition

We first provide relevant notations in figure 2. These notations will be used throughout the paper. The data placement problem for wireless broadcasting is to find a broadcast schedule σ that minimizes the total access time (*TAT*), denoted by:

$$TAT(\sigma) = \sum_{q_i \in \mathcal{Q}} AT^{avg}(q_i, \sigma) \times freq(q_i),$$

where $AT^{avg}(q_i, \sigma)$ is the average access time of the query q_i based on σ.

The access time of a query depends on the starting time of the query as shown in figure 1. If, for example, the query is submitted at the beginning of the current *bcast*, then both the data objects $d1$ and $d4$ can be retrieved in a single *bcast*. Thus we need to derive the formula for the average access time of a query, $AT^{avg}(q_i, \sigma)$.

Figure 3 shows a data set of a query q_i being broadcasted, where $QDS(q_i) = \{d_1, d_2, \ldots, d_n\}$ and n is the number of data objects that q_i accesses. The interval between two data objects d_j and d_{j+1} is denoted as δ_j which is $t_j - |d_j|$, where t_j is the length (in terms of packets) from the start of d_j to the start of the next data object, d_{j+1}. Because the broadcasting is repeated for a given duration, the total summation of t_j i.e., $\sum_{j=1}^{j=n} t_j$, is the size of one bcast B.

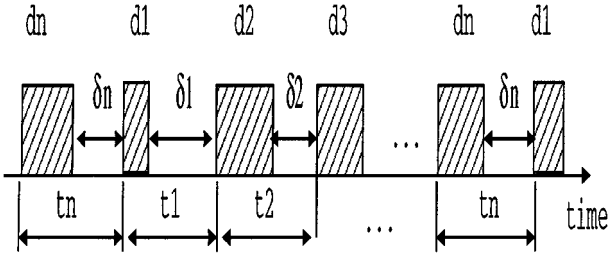

Figure 3. Placement of a query data set.

Let us assume that $p(x)$ and $F(x)$ are the probability and the access time of q_i issued at location x respectively. Then the estimated average access time of q_i is computed as follows.

$$\int_0^B p(x) \cdot F(x)\, dx = \frac{1}{B} \int_0^B F(x)\, dx \tag{1}$$

$$= \frac{1}{B} \sum_{j=1}^{n} \int_0^{t_j} F(y)\, dy \tag{2}$$

$$= \frac{1}{B} \sum_{j=1}^{n} \left[\int_0^{|d_j|} F(y)\, dy + \int_{|d_j|}^{t_j} F(y)\, dy \right] \tag{3}$$

$$= \frac{1}{B} \sum_{j=1}^{n} \left[\int_0^{|d_j|} \kappa B\, dy + \int_{|d_j|}^{t_j} \kappa (B - y + |d_j|)\, dy \right] \tag{4}$$

$$= \kappa B - \frac{1}{2B} \sum_{j=1}^{n} \kappa (t_j - |d_j|)^2 \tag{5}$$

$$= \kappa \left(B - \frac{1}{2B} \sum_{j=1}^{n} \delta_j^2 \right) \tag{6}$$

The estimated average access time is computed by summing up $p(x) \times F(x)$ for all location x during one *bcast* and $p(x)$ is $\frac{1}{B}$ regardless of x(1). It can be computed by dividing the *bcast* into t_j, the interval between each data object (2). The interval t_j is divided into two smaller intervals, $0 \sim |d_j|$ and $|d_j| \sim t_j$ (3).

In the interval $0 \sim |d_j|$, the access time $F(y)$ is the period of one *bcast* because parts of d_j have passed over. The client has to read the remaining parts of data at the next *bcast* and it takes κB with respect to time, where κ is the factor converting data size into time dimension (4). In the interval $|d_j| \sim t_j$, the access time is the period of one *bcast* minus the interval $y - |d_j|$, where y is the probe position in the unit interval (4). In consequence the average access time is computed as in (6).

The measure of average access time, which we derived in the above, is still difficult for us to develop data scheduling method for wireless broadcasting because the formula is an

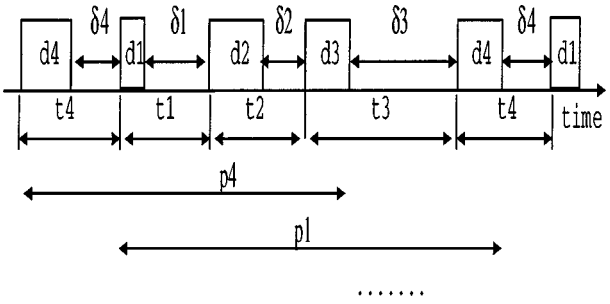

Figure 4. Graphical illustration of the QD of a query.

equation of the second degree. Now we define a new measure, called *Query Distance*(*QD*), that indicates the degree of coherence of the data objects a query accesses.

Definition 1. Suppose $QDS(q_i)$ is $\{d_1, d_2, \ldots, d_n\}$, and δ_j is the interval between d_j and d_{j+1} in schedule σ. Then the *Query Distance* (*QD*) of q_i in σ is defined as:

$$QD(q_i, \sigma) = B - MAX(\delta_k), \quad k = 1 \sim n.$$

□

The measure of QD can be interpreted like in figure 4. When a $QDS(q)$ is comprised of $\{d1, d2, d3, d4\}$, the QD of q in the given schedule is equivalent to the minimal one among p_1, p_2, p_3, and p_4, where p_i is the access time when the query starts at the beginning of d_i.

Lemma 1. *Given a query q_i and two schedules σ_1 and σ_2,*

$$\text{if } QD(q_i, \sigma_1) \geq QD(q_i, \sigma_2) \text{ then } AT^{avg}(q_i, \sigma_1) \geq AT^{avg}(q_i, \sigma_2)$$

Proof: Omitted for brevity. See the reference [3]. □

Let the total query distance denoted by $TQD(\sigma)$ is defined as $\sum_{q_i \in \mathcal{Q}} QD(q_i, \sigma) \times freq(q_i)$. Now we redefine the problem of wireless data placement with the measure *QD* based on Lemma 1.

Definition 2. Given a set of data objects $\mathcal{D}$ and a set of queries $\mathcal{Q}$, the wireless data placement problem is to find a broadcast schedule σ_i such that $TQD(\sigma_i)$ is minimum among all possible σ_i, $i = 1, \ldots$. □

Theorem 1. *The wireless data placement problem in Definition* 2 *is NP-complete.*

Proof: The proof follows transformation from Optimal Linear Arrangement Problem (OLAP) [6] that is known as a NP-complete problem. OLAP is to find a one-to-one mapping function f such that $f : V \rightarrow \{1, 2, \ldots, |v|\}$, $\sum_{\{u,v\} \in E} |f(u) - f(v)| \leq k$ in a

graph $G = (V, E)$, where k is a positive constant. Let's assume that each query accesses only two data objects, the size of each data object is equal and the reference frequency of each query is equal. Then the problem is transformed into OLAP by considering the data object as the node in the graph and the query as the edge in the graph. For details, see the reference [3]. □

3. The proposed data placement method

In this section we propose a data placement method for wireless broadcast. After explaining the basic idea of our method with a simple example, we formally describe the method.

3.1. Basic idea

Now we take a simple example and show our basic idea. Let us assume that there are eight data objects to be broadcasted and three queries that mobile clients submit onto the broadcasting channel. The data set of each query (i.e., *QDS*) is depicted as in figure 5. All data objects are assumed to be equal (in size) and the occurrence frequency of each query is assumed as follows:

$$freq(q_1) = 3, \qquad freq(q_2) = 2, \qquad freq(q_3) = 1$$

Initially the schedule σ^{step0} is empty. The method finds the highest frequency query, q_1, and expands σ^{step0} with $QDS(q_1)$. Then current schedule σ^{step1} becomes

$$\sigma^{step1} = [d_1, d_2, d_4, d_5].$$

Data objects in σ^{step1} are interchangeable one another, for $QD(q_1, \sigma^{step1})$ is the same with any permutation of d_1, d_2, d_4 and d_5. We use the symbols '[' and ']'. The data objects bounded by a pair of square brackets, '[' and ']', are interchangeable one another without changing the *TQD* value. For notational convenience, we will not use angle brackets ($\langle$ and $\rangle$) for intermediate schedules, if there is no ambiguity. That is, we use angle brackets for only final schedule.

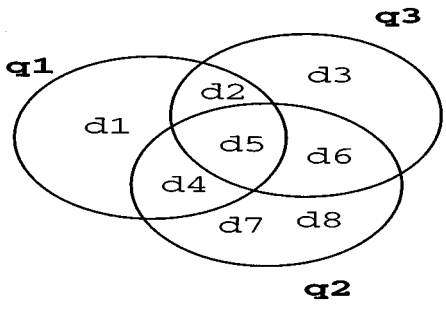

Figure 5. Queries and their QDS's.

Next, the algorithm considers query q_2, whose *QDS* is $\{d_4, d_5, d_6, d_7, d_8\}$. Since current schedule σ^{step1} contains the data objects d_4 and d_5 that are in $QDS(q_2)$, the schedule can be expanded into one of the following two forms:

$$\sigma^{step2}_{RightAppend} = [d_1, d_2][d_4, d_5][d_6, d_7, d_8]$$
$$\sigma^{step2}_{LeftAppend} = [d_6, d_7, d_8][d_4, d_5][d_1, d_2].$$

The former schedule is the result of appending $QDS(q_2)$ at the right end of σ^{step1} whereas the latter one is that of left appending. As data objects bounded by a pair of square brackets are freely interchangeable, there are $24(= 2 * 2 * 6)$ possible ways of data ordering for each schedule. However, all of them give the same *TQD* value. In this example we choose the former for further expanding process. The schedule σ^{step2} minimizes the *QD* of q_2 while preserving the *QD* of q_1 i.e., $QD(q_1, \sigma^{step1}) = QD(q_1, \sigma^{step2})$.

Finally $QDS(q_3)$ is processed. Among data objects in $QDS(q_3)$ ($\{d_2, d_3, d_5, d_6\}$), only d_3 is not included in the current schedule σ^{step2}. Insertion of d_3 into any position of σ^{step2} increases the *QD* values of q_1 and(or) q_2. Our method gives priority to the higher-frequency queries when minimizing *QD* i.e., the *QD* of a query must not be increased by a lower-frequency query. Hence d_3 has to be appended to the left or right end of σ^{step2}. When appending d_3, data objects of d_2, d_5, and d_6 need to be moved (if allowed) for minimizing the *QD* of q_3 like follows.

$$\sigma^{step3}_{LeftAppend} = [d_3][d_1, d_2][d_4, d_5][d_6][d_7, d_8]$$
$$\sigma^{step3}_{RightAppend} = [d_1][d_2][d_4, d_5][d_6, d_7, d_8][d_3]$$

In two schedules above, $\sigma^{step3}_{LeftAppend}$ gives smaller *TQD*, for $QD(q_3, \sigma^{step3}_{LeftAppend})$ is less than $QD(q_3, \sigma^{step3}_{RightAppend})$ and those of q_1 and q_2 are equal. In consequence the final schedule becomes one of the forms in the below, which are results of $\sigma^{step3}_{LeftAppend}$ by possible interchanging data objects.

$\langle d_3, d_1, d_2, d_4, d_5, d_6, d_7, d_8\rangle$
or $\langle d_3, d_2, d_1, d_4, d_5, d_6, d_7, d_8\rangle$
or $\langle d_3, d_1, d_2, d_5, d_4, d_6, d_7, d_8\rangle$
or $\langle d_3, d_2, d_1, d_4, d_5, d_6, d_8, d_7\rangle$
and so on.

After describing a few properties related with the wireless data schedule, we explain the proposed scheduling method in more detail.

Property 1 (Interchangeability)**.** *Given a query q_i and two schedules σ_1 and σ_2 such that*

$$\sigma_1 = \langle d_1, d_2, \ldots, d_{i-1}, d_i, d_{i+1}, \ldots, d_{j-1}, d_j, d_{j+1}, \ldots, d_{N-1}, d_N\rangle$$
$$\sigma_2 = \langle d_1, d_2, \ldots, d_{i-1}, d_j, d_{i+1}, \ldots, d_{j-1}, d_i, d_{j+1}, \ldots, d_{N-1}, d_N\rangle,$$

if $d_i, d_j \in QDS(q_i)$ then $QD(q_i, \sigma_1) = QD(q_i, \sigma_2)$ □

schedule ::= segment < '⊕' segment >

segment ::= fragment < fragment >

fragment ::= '[' component ']'

component ::= element < ',' element >

element ::= data | '[' component ']'

Figure 6. Composition of the broadcast data schedule.

This property is a direct consequence from the definition of *Query Distance*. The property says that any two data objects in a $QDS(q_i)$ are mutually interchangeable with respect to q_i.

Definition 3. Given two schedules σ_1 and σ_2, if $QD(q_i, \sigma_1)$ is equal to $QD(q_i, \sigma_2)$ for all queries q_i, then we call σ_1 is equivalent to σ_2, denoted by $\sigma_1 \equiv \sigma_2$. □

Property 2 (Symmetry)**.** *If a schedule σ_2 is an inverse of σ_1, that is*

$$\sigma_1 = \langle d_1, d_2, \ldots, d_{i-1}, d_i, d_{i+1}, \ldots, d_{N-1}, d_N \rangle$$
$$\sigma_2 = \langle d_N, d_{N-1}, \ldots, d_{i+1}, d_i, d_{i-1}, \ldots, d_2, d_1 \rangle,$$

then σ_1 is equivalent to σ_2 ($\sigma_1 \equiv \sigma_2$). □

Since for all j, δ_j in σ_1 is equal to that in σ_2, $QD(q_i, \sigma_1)$ is equal to $QD(q_i, \sigma_2)$ for all q_i.

In the method the data broadcast schedule is composed of a few constructs which are described in figure 6 based on EBNF (Extended Backus-Naur Form) methodology. This constructs are internally used in our method and not parts of real broadcast schedule. A schedule consists of segments(s_i) that are connected by '⊕' symbol. The ordering of segments are flexible within a schedule. If, for example, a schedule σ is "$s_1 \oplus s_2 \oplus s_3$", then the schedules "$s_1 \oplus s_3 \oplus s_2$", ..., "$s_2 \oplus s_1 \oplus s_3$" are all equivalent. The fragment(f_i) is composed of a component wrapped by '[' and ']' and the ordering of fragments are fixed in the given schedule. The element (e_i), which constitutes a component, is made up of (a) data or another component wrapped by '[' and ']'. Any two constituents within '[' and ']' can be freely interchanged with each other by Property 1.

Example 1.

$$\begin{aligned}\sigma &= \overbrace{[d_1, d_2][d_3][d_4, d_5, [d_6, d_7], d_8]}^{s_1} \oplus \overbrace{[d_9, [d_{10}, d_{11}], d_{12}]}^{s_2} \oplus \overbrace{[d_{13}]}^{s_3} \\ &\equiv [d_2, d_1][d_3][d_5, d_8, [d_7, d_6], d_4] \oplus [d_{13}] \oplus [[d_{10}, d_{11}], d_{12}, d_9] \\ &\equiv [d_{12}, d_9, [d_{11}, d_{10}]] \oplus [d_{13}] \oplus [d_2, d_1][d_3][[d_7, d_6], d_4, d_5, d_8]\end{aligned}$$

$\equiv$ and so on.

$$\sigma' = \underbrace{[d_3][d_1, d_2]}[d_4, d_5, [d_6, d_7], d_8] \oplus [d_9, [d_{10}, d_{11}], d_{12}] \oplus [d_{13}]$$

The schedule σ consists of 3 segments. s_1 has 3 fragments, and s_2 and s_3 has one fragment each. The third fragment of s_1 has four elements: d_4, d_5, $[d_6, d_7]$ and d_8. And the element $[d_6, d_7]$ has two elements, d_6 and d_7, in nested style. The order of the elements within '[' and ']' is not fixed whereas that of the fragments is fixed. The schedule σ' is not equivalent to σ because the order of fragments in s_1 is changed. □

3.2. Schedule adjustment operators

Our method constructs broadcast schedule by inserting *QDS* of each query based on greedy strategies, that is, the *QD* of higher frequency query is guaranteed not to be changed when minimizing *QD* of the lower one. For this purpose we define five basic operations *MRF*, *MLF*, *MRS*, *MLS* and *MDC*. Each of them is used when expanding the given schedule (a fragment or a segment) by appending a QDS.

The operation *MRF*(Move data to the Right of a Fragment) and *MRS*(Move data to the Right in a Segment) move the data objects, which are included in both the schedule and the *QDS*, to the right as far as possible when appending a *QDS* to the right end of the given schedule. This is for minimizing the *QD* of the currently considered query. *MLF* (Move data to the Left of a Fragment) and *MLS* (Move data to the Left in a Segment) are symmetric operations of *MRF* and *MRS* respectively. They are used when appending a *QDS* to the left end of the given schedule. The operation *MDC* (Move Data Closest) is used when all data objects in *QDS* are included in the current schedule. By operation *MDC* the data objects, which are included in the *QDS* and the given schedule, are moved closest for the purpose of minimizing *QD*.

Definition 4.
MRF(f_i, *QDS*)

1. In the given fragment f_i find the elements e_i such that $DS(e_i) \subset QDS$. Then remove them from f_i and make them as a new fragment f_γ.
2. In f_i find the elements e_j such that $DS(e_j) \cap QDS \neq \emptyset$. Among them select the element whose Ω value is maximum and form it into a new fragment f_α and the others into a new fragment f_β after being removed from f_i.

$$\Omega_{e_j} = \sum_{d_i \in e_j \wedge \ not \ in \ QDS} |d_i|$$

3. When assuming that f_i' is f_i from which the elements e_i and e_j have been removed, the result of *MRF*(f_i, *QDS*) has the form of:[2]

$$f_i' \ MRF(f_\alpha, DS(f_\alpha) \cap QDS) \ f_\beta \ f_\gamma.$$

MRS(s_i, QDS)

1. Among the fragments in s_i, find the left-most fragment f_i such that $DS(f_i) \cap QDS \neq \emptyset$.
2. Order the fragments of s_i[3] like this:

$$s_i = \cdots f_{i-1}\, MRF(f_i, DS(f_i) \cap QDS)\, f_{i+1} \cdots$$

MDC(s_i, QDS)

1. Find the left-most and right-most fragments (respectively f_L and f_R) in s_i such that $DS(f_L) \cap QDS \neq \emptyset$ and $DS(f_R) \cap QDS \neq \emptyset$.
2. If $f_L \neq f_R$, then order the fragments of s_i as follows.

$$\begin{aligned} s_i = \cdots & f_{L-1}\, MRF(f_L, DS(f_L) \cap QDS)\, f_{L+1} \cdots \\ \cdots & f_{R-1}\, MLF(f_R, DS(f_R) \cap QDS)\, f_{R+1} \cdots \end{aligned}$$

3. Otherwise, $s_i = \cdots f_{L-1}\, MLF(f_L, DS(f_L) \cap QDS)\, f_{R+1} \cdots$ □

Example 2. Suppose the sizes of all data objects are equal. Let fragment f_i, segment s_i and the QDS's of the given queries be formed as follows:

$$\begin{aligned} f_i &= [d_1, d_2, [d_3, d_4], d_5, [d_6, [d_7, d_8], d_9]], \\ s_i &= [d_1, d_2][d_3, d_4][d_5, [d_6, [d_7, d_8], d_9]], \\ [QDS_1] &= \{d_2, d_4, d_6, d_8, d_{10}\}, QDS_2 = \{d_2, d_4, d_6, d_8\}. \end{aligned}$$

Then the results of the basic operations in Definition 4 are as follows:

$$\begin{aligned} MRF(f_i, QDS_1) &= [d_1, d_5][d_9][d_7][d_8][d_6][d_3, d_4][d_2] \\ MRS(s_i, QDS_1) &= [d_1][d_2][d_3, d_4][d_5, [d_6, [d_7, d_8], d_9]] \\ MDC(s_i, QDS_2) &= [d_1][d_2][d_3, d_4][d_6][d_8][d_7][d_9][d_5]. \end{aligned}$$

We explain the steps of $MRF(f_i, QDS_1)$. In Step 1 find the elements (e_j) that are totally included in the QDS_1: d_2. In Step 2, find the elements (e_j): $[d_3, d_4]$ and $[d_6, [d_7, d_8], d_9]$. In the elements (e_j), the Ω value of the latter is greater than that of the former. Thus, $f_i' = [d_1, d_5]$, $f_\alpha = [d_6, [d_7, d_8], d_9]$, $f_\beta = [d_3, d_4]$ and $f_\gamma = [d_2]$. In recursive manner $MRF(f_\alpha, \{d_6, d_8\})$ becomes $[d_9][d_7][d_8][d_6]$. □

Lemma 2 (Preservation**).** *The basic operations* (*in Definition* 4) *preserve the QD's of the queries that have been previously used for constructing the schedule.*

Proof: For each operation the elements in a fragment can be interchanged one another and also can be moved into a new fragment. But in any cases no new element

is inserted between them. So the *QD's* of the previously used queries remain unchanged when expanding the schedule with a new *QDS*. In Example 2, if there are some queries whose *QDS*'s are $\{d_3, d_4\}$, $\{d_7, d_8\}$, and $\{d_6, d_7, d_8, d_9\}$ respectively, then their *QD*'s remain unchanged in the result of $MRF(f_i, QDS_1)$, $MRS(f_i, QDS_1)$ and $MDC(s_i, QDS_2)$. □

3.3. *The algorithm*

The proposed method constructs broadcast schedule by expanding the schedule with *QDS* of each query in greedy manner after sorting the queries based on $freq(q_i)$. The algorithm of this method is described in figure 7 where schedule expansion rules are described later. The basic properties of our algorithm are as follows:

1. The higher-frequency query takes precedence over the lower-one. In other words, minimizing the *Query Distance* of a query with higher frequency needs to be considered more seriously than that of a query with lower frequency.
2. When expanding the schedule with a query, the *QD*'s of the queries that have been previously processed remain unchanged.
3. When expanding the schedule with query q_i, the proposed method always minimizes the *QD* of q_i as much as possible.

Schedule Expansion Rules: When expanding schedule σ with $QDS(q_i)$, each schedule σ' below minimizes $QD(q_i)$ while preserving the *QD*'s of the higher-frequency queries i.e., the queries that have been previously used for constructing σ. Let us assume that

$$\sigma = s_1 \oplus \cdots \oplus s_k.$$

1. when $DS(s_i) \cap QDS(q_i) = \emptyset$ for all s_i, make a new segment s_{k+1} with the data in *QDS* (q_i) and set "$\sigma' = \sigma \oplus s_{k+1}$".

Input: a set of data objects D, and a set of queries Q

Output: a broadcast schedule σ

Method:

1. Initially σ is empty.
2. Sort the queries in non-increasing order of frequencies.
3. For each query q_i in the sorted order,

   expand σ with $QDS(q_i)$ by using **Schedule Expansion Rules.**

Figure 7. Algorithm description.

2. when there is one segment s_i such that $DS(s_i) \cap QDS(q_i) \neq \emptyset$, the result becomes

$$\sigma' = \cdots \oplus s_{i-1} \oplus s_i' \oplus s_{i+1} \oplus \cdots,$$

where s_i' is constructed as follows.

(a) if $QDS(q_i)$ is included in $DS(s_i)$, then s_i' is the result of $MDC(s_i, QDS(q_i))$.
(b) if $DS(s_i)$ is included in $QDS(q_i)$, then transform s_i into s_i' with the following form:
   i. when s_i consists of one fragment:
   $$s_i' = [s_i, QDS(q_i) - DS(s_i)]$$
   ii. when s_i consists of more than one fragment:[4]
   $$s_i' = s_i[QDS(q_i) - DS(s_i)]$$
(c) otherwise, transform s_i into s_i' as one of the following whose $QD(q_i, \sigma_x)$ is not larger than the other.

$$\sigma_{RightAppend} = MRS(s_i, QDS(q_i))[QDS(q_i) - DS(s_i)]$$
$$\sigma_{LeftAppend} = [QDS(q_i) - DS(s_i)]\, MLS(s_i, QDS(q_i))$$

3. when the data of $QDS(q_i)$ are spread over two segments s_i and s_j, construct σ' as

$$\sigma' = \cdots \oplus s_{i-1} \oplus s_i' \oplus s_{i+1} \oplus \cdots \oplus s_{j-1} \oplus s_{j+1} \oplus \cdots,$$

where s_i' is organized as follows.

(a) if all the data objects in both the segments are included in $QDS(q_i)$, then two segments are transformed into one segment s_i' with this form:
   i. when s_i and s_j consist of one fragment:
   $$s_i' = [s_i, s_j, QDS(q_i) - (DS(s_i) \cup DS(s_j))]$$
   ii. when only s_i consists of one fragment:
   $$s_i' = s_j[s_i, QDS(q_i) - (DS(s_i) \cup DS(s_j))]$$
   iii. when only s_j consists of one fragment:
   $$s_i' = s_i[s_j, QDS(q_i) - (DS(s_i) \cup DS(s_j))]$$
   iv. when s_i and s_j consist of more than one fragment:
   $$s_i' = s_i s_j[QDS(q_i) - (DS(s_i) \cup DS(s_j))]$$
(b) if all the data objects in only one segment s_i is totally included in $QDS(q_i)$, then combine s_i and s_j and make new segment s_i' with following form:
   i. when s_i consists of one fragment:
   $$s_i' = [s_i, QDS(q_i) - (DS(s_i) \cup DS(s_j))]\, MLS(s_j, QDS(q_i))$$
   ii. when s_i consists of more than one fragment:
   $$s_i' = s_i, [QDS(q_i) - (DS(s_i) \cup DS(s_j))]\, MLS(s_j, QDS(q_i))$$

(c) if $DS(s_i)$ and $DS(s_j)$ are not included in $QDS(q_i)$, then, by combining s_i and s_j, form a new segment s_i' whose $QD(q_i, \sigma_x)$ is not larger than the others among the following schedules:

$$\sigma_{LeftAppend} = [QDS(q_i)']s_i MLS(s_j, QDS(q_i)),$$
$$\sigma_{Interposition} = MRS(s_i, QDS(q_i))[QDS(q_i)'] MLS(s_j, QDS(q_i)),$$
$$\sigma_{RightAppend} = MRS(s_i, QDS(q_i))s_j[QDS(q_i)'],$$

where $QDS(q_i)'$ is $QDS(q_i) - (DS(s_i) \cup DS(s_j))$.

4. when the data objects in $QDS(q_i)$ are spread over more than two segments, do the following steps.

- Classify the segments into two groups: the segments s_i^s such that $QDS(q_i) \cap DS(s_i^s) \neq \emptyset$ and the segments s_j^d such that $QDS(q_i) \cap DS(s_j^d) = \emptyset$
- Among the segments s_i^s, find two segments whose Ψ values are the largest s_{1st}^s and the second largest s_{2nd}^s.

$$\Psi_{s_i} = \sum_{k=1}^{\omega-1} |data(MRS(s_i, QDS(q_i)), k)|,$$

where $\omega = Min(loc(d_j, MRS(s_i, QDS(q_i))))$ for the data $d_j \in QDS(q_i)$.

- Make the schedule as follows.

$$\sigma' = \sigma_1 \oplus \sigma_2,$$
$$\sigma_1 = \cdots \oplus s_j^d \oplus \cdots,$$
$$\sigma_2 = MRS\big(s_{1st}^s, QDS(q_i)\big)\sigma_3\, MLS\big(s_{2nd}^s, QDS(q_i)\big),$$
$$\sigma_3 = s_i'^s[QDS(q_i)'],$$

where $s_i'^s$ is s_i^s from which s_{1st}^s and s_{2nd}^s are removed and $QDS(q_i)' = QDS(q_i) - \bigcup s_i^s$. □

Example 3. Let us take examples of using Schedule Expansion Rules. We give an input schedule σ, a query's data set QDS and its expanding result σ' for each case by case of the rules.

1.

$$\sigma = [d_4, [d_5, d_6], d_7] \oplus [d_8, d_9]$$
$$QDS = \{d_1, d_2, d_3\}$$
$$\sigma' = [d_4, [d_5, d_6], d_7] \oplus [d_8, d_9] \oplus [d_1, d_2, d_3]$$

2. (a)

$$\sigma = [d_1, d_2][d_4, [d_5, d_6], d_7] \oplus [d_8, d_9]$$
$$QDS = \{d_1, d_4, d_5\}$$
$$\sigma' = [d_2][d_1][d_4][d_5][d_6][d_7] \oplus [d_8, d_9]$$

(b) i.

$$\begin{aligned}\sigma &= [d_4, [d_5, d_6], d_7] \oplus [d_8, d_9] \\ QDS &= \{d_8, d_9, d_{10}\} \\ \sigma' &= [d_4, [d_5, d_6], d_7] \oplus [[d_8, d_9], d_{10}]\end{aligned}$$

ii.

$$\begin{aligned}\sigma &= [d_4, [d_5, d_6], d_7] \oplus [d_8][d_9] \\ QDS &= \{d_8, d_9, d_{10}\} \\ \sigma' &= [d_4, [d_5, d_6], d_7] \oplus [d_8][d_9][d_{10}]\end{aligned}$$

(c)

$$\begin{aligned}\sigma &= [d_4, [d_5, d_6], d_7] \oplus [d_8, d_9] \\ QDS &= \{d_8, d_{10}\} \\ \sigma' &= [d_4, [d_5, d_6], d_7] \oplus [d_9][d_8][d_{10}]\end{aligned}$$

The Right Append result, $[d_{10}][d_8][d_9]$, is symmetric one of this, so it is equivalent by Property 2.

3. (a) i.

$$\begin{aligned}\sigma &= [d_4, [d_5, d_6], d_7] \oplus [d_8] \\ QDS &= \{d_4, d_5, d_6, d_7, d_8, d_9, d_{10}\} \\ \sigma' &= [[d_4, [d_5, d_6], d_7], d_8, d_9, d_{10}]\end{aligned}$$

ii.

$$\begin{aligned}\sigma &= [d_4][[d_5, d_6], d_7] \oplus [d_8] \\ QDS &= \{d_4, d_5, d_6, d_7, d_8, d_9, d_{10}\} \\ \sigma' &= [d_4][[d_5, d_6], d_7][d_8, d_9, d_{10}]\end{aligned}$$

iii. Similar to the above case.

iv.

$$\begin{aligned}\sigma &= [d_4][d_5, d_6] \oplus [d_7][d_8] \\ QDS &= \{d_4, d_5, d_6, d_7, d_8, d_9, d_{10}\} \\ \sigma' &= [d_4][d_5, d_6][d_7][d_8][d_9, d_{10}]\end{aligned}$$

(b) i.

$$\begin{aligned}\sigma &= [d_4, [d_5, d_6], d_7] \oplus [d_8, d_9] \\ QDS &= \{d_7, d_8, d_9, d_{10}\} \\ \sigma' &= [[d_8, d_9], d_{10}][d_7][d_4, [d_5, d_6]]\end{aligned}$$

ii.

$$\begin{aligned}\sigma &= [d_4, [d_5, d_6], d_7] \oplus [d_8][d_9] \\ QDS &= \{d_7, d_8, d_9, d_{10}\} \\ \sigma' &= [d_8][d_9][d_{10}][d_7][d_4, [d_5, d_6]]\end{aligned}$$

(c)

$$\begin{aligned} \sigma &= [d_4, [d_5, d_6], d_7] \oplus [d_8, d_9] \\ QDS &= \{d_7, d_8, d_{10}\} \\ \sigma' &= [d_4, [d_5, d_6]][d_7][d_{10}][d_8][d_9] \end{aligned}$$

In this case the result of Interposition gives the smallest *QD* value.

4.

$$\begin{aligned} \sigma &= [d_1] \oplus [d_4, [d_5, d_6], d_7] \oplus [d_8, d_9] \oplus [d_{10}] \\ QDS &= \{d_3, d_7, d_8, d_{10}\} \\ \sigma' &= [d_1] \oplus [d_4, [d_5, d_6]][d_7][d_{10}][d_3][d_8][d_9] \end{aligned}$$

□

4. Performance evaluation

The running overhead of our method is determined by the Step 2 and 3 of the algorithm in figure 7. In Step 2 the complexity is O($MlogM$) where M is the number of query patterns. Step 3 shows the complexity of O(MN) where N is the number of data objects, for the schedule expansion process searches all or parts of the data objects in the schedule for each query. The method can be processed with CPU speed, so we do not consider its running overhead in the experiment.

In wireless environment the server periodically analyzes the queries' data access patterns and reorganizes the broadcast data stream. In the experiment we consider the placement of the broadcast data in a single period.

There is no work on the data placement in the environment: a query accesses multiple data objects and there is no replication in a single *bcast*. We measure the performance improvement i.e., *TAT*(*Total Access Time*) reduction of the proposed method against sequential (with respect to data id.) schedules on which no effort of data placement is put. The experimental results are mean values over more than 20 tries based on *Central Limit Theorem* [5]. The related parameters are as follows.

The number of data objects that are delivered on broadcasting channel is N. Every data object on broadcast channel is accessed by one or more queries. The number of query patterns that access parts of the broadcast data set is M. Every query accesses at least one data object on the broadcasting channel and the *QDS*'s are mutually independent between queries. *Selectivity* is the degree of a query's *QDS* size over the size of broadcast data set in terms of percentage. For example 2% selectivity means a query accesses 2% of the broadcast data set. We assume the *QDS* of each query is uniformly distributed over the broadcast data set. We consider three different kinds of distribution of query's occurrence frequency: uniform distribution, normal distribution, and exponential distribution ($\lambda = 1$). In all the three distributions we set the minimum frequency into 1 and maximum one into M. We set the size of one data object as 10 packet length. The packet is basic unit of data manipulation on wireless channel. We experiment with several other values, but the results are almost equal. We assume the size of each data object to be equal.

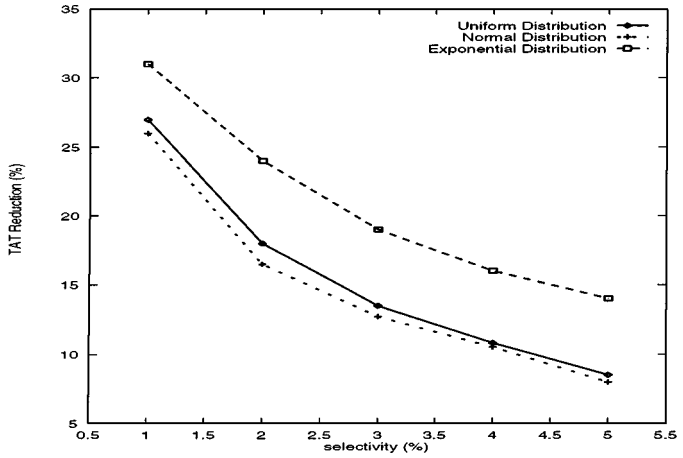

Figure 8. *TAT* reduction with change in selectivity.

In the first experiment we change selectivity values with 1000 data objects and 100 queries. The result of the experiment is shown in figure 8. The performance improvement increases to more than 25%–30% when the selectivity is low and decreases to about 10% when the selectivity is high. And the case of exponential distribution of $freq(q_i)$ provides better performace than the others. The result shows that more highly skewed query distributions achieve better performance with our method. It is because our method adopts greedy philosophy, that is, it gives priority to the higher-frequency queries when reducing *QD*. In the following experiments we use exponential distribution for query's occurrence frequency.

In the second experiment we change the number of data objects (N) with 100 queries and 2% selectivity. The result of this experiment is in figure 9. As shown in this result, the

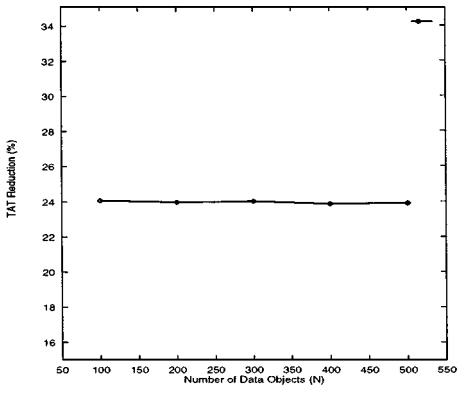

Figure 9. *TAT* reduction with change in the number of data objects.

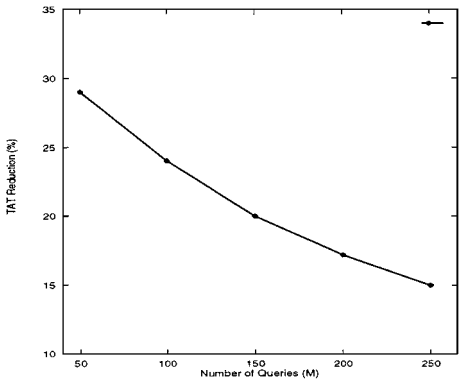

Figure 10. *TAT* reduction with change in the number of query patterns.

performance of the proposed method has little dependency on the number of data objects. This means that the method is usable for broadcasting lots of data objects as in real wireless environment.

The third experiment changes the number of queries while fixing N (1000) and selectivity (2%). The result in figure 10 shows the performance improvement decreases with large number of queries.

In above experiments we can observe that the performance improvement of our method decreases with large number of queries or high selectivity. It is because the *QD* of a query gets longer when its *QDS* is overlapped much with those of others. Based on this fact we define a parameter $\mathcal{DOD}$ (Degree Of Duplication: %) such that

$$\mathcal{DOD} = \text{Selectivity} * \text{Number of Query Patterns } (M).$$

Figures 8 and 10 are based on same $\mathcal{DOD}$ variation, in spite of different M's and selectivities, so show almost similar improvement patterns. In other words the performance of our method is dependent on $\mathcal{DOD}$ essentially.

However, when comparing figure 8 with figure 10, we can see that two cases of the same $\mathcal{DOD}$ do not give exactly the same improvement. The case of low selectivity provides a little better performace than the other. For example the case of 100 queries with 1% selectivity (in figure 8) gives better result than that of 50 queries with 2% (in figure 10) regardless of the same 100% $\mathcal{DOD}$.

5. Conclusion

In this paper we have introduced the data placement problem for wireless broadcast when user queries access more than one data object on wireless broadcast channel. As discussed, how to schedule the broadcast data can affect access time significantly.

There are some relevant works on scheduling broadcast data. However, as far as we are aware of, no work considers the case when a query accesses more than one data object, which is common in practice.

We have defined the problem of wireless data placement and a new measure named *Query Distance*, and then proposed a heuristic method using the measure.

Performance of the proposed method has been experimented with various parameters. According to the result we observe the proposed method effectively construct wireless broadcast schedule and reduce the access time of mobile query.

As a further work, we will explore the data placement method for multi-channel environment and non-uniform broadcasting environment, that is, the case data objects are replicated in a single broadcast.

Notes

1. In this work we assume there is no precedence relation in accessing data objects.
2. In case of *MLF*, the fragments are arranged like "$f_\gamma\, f_\beta MLF(f_\alpha, DS(f_\alpha) \cap QDS)\, f_i'$".
3. In case of *MLS*, the fragments are arranged like "$\cdots f_{i-1} MLF(f_i, DS(f_i) \cap QDS)\, f_{i+1} \cdots$" where f_i is the right-most fragment in s_i such that $DS(f_i) \cap QDS \neq \emptyset$.
4. "s_j[*some schedule*]" is short for "$f_1 f_2 \cdots f_k$[*some schedule*]" where f_i $(i = 1, \ldots, k)$ is a fragment of s_j.

References

1. S. Acharya, R. Alonso, M. Franklin, and S. Zdonik, "Broadcast disks: Data management for asymmetric communication environments," in Proceedings of ACM SIGMOD Conference, 1995, pp. 199–210.
2. S. Acharya, M. Franklin, and S. Zdonik, "Disseminating updates on broadcast disks," in Proceedings of Very Large Data Bases Conference, 1996, pp. 354–365.
3. Y.D. Chung and M.H. Kim, "On scheduling wireless broadcast data," Technical Report CS-TR-98-134, KAIST, Department of Computer Science, 1998.
4. Y.D. Chung and M.H. Kim, "An index replication scheme for wireless data broadcasting," Journal of Systems and Software, vol. 51, no. 3, 2000, pp. 191–199.
5. J.L. Devore, Probability and Statistics for Engineering and the Sciences, 3 edition, Brooks/Cole Publishing Company, 1990.
6. M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman Publishing Company, 1976.
7. T. Imielinski, S. Viswanathan, and B.R. Badrinath, "Data on air: Organization and access," IEEE Transactions on Knowledge and Data Engineering, vol. 9, no. 3, 1997, pp. 353–372.
8. T. Imielinski, S. Viswanathan, and B.R. Badrinath, "Energy efficient indexing on air," in Proceedings of ACM SIGMOD Conference, 1994, pp. 25–36.
9. C. Su, L. Tassiulas, and V.J. Tsotras, "Broadcast scheduling for information distribution," Wireless Networks, vol. 5, no. 2, 1998, pp. 137–147.
10. K. Tan and J.X. Yu. "Generating broadcast programs that support range queries," IEEE Transactions on Knowledge and Data Engineering, vol. 10, no. 4, 1998, pp. 668–672.