QEM: A Scheduling Method for Wireless Broadcast Data

Yon Dohn Chung and Myoung Ho Kim
Department of Computer Science
Korea Advanced Institute of Science and Technology
373-1, Kusung-dong, Yusung-gu, Taejon, 305-701, Korea

Abstract

In mobile distributed systems the data on air can be accessed by a large number of clients. This paper describes the way clients access the wireless broadcast data in short latency. We define and analyze the problem of wireless data scheduling. And we propose a measure, named QueryDistance(QD), which represents the degree of coherence for the data set accessed by a query. We give a practically usable method named QEM which constructs the broadcast schedule by expanding each query's data set in greedy way. We also evaluate the performance of our method by experiments.

1. Introduction

In mobile environments there are two kinds of communication modes: one-way mode and two-way mode. The first is that server system broadcasts data in public channel and clients just listen to the channel. The other is that clients send request messages to the server and then receive the reply messages from the server. Due to the restriction of the client's energy usage two-way mode is recommended not to be used frequently.

There are two parameters which are related to data broadcasting. They are access time and tuning time. Access time is the time elapsed from the moment a client submits a query to the receipt of the data of his(her) interest on the broadcast channel. 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, nonuniform broadcasting, and those on reducing tuning time such as indexing, hashing and so on. In this work we are investigating a data scheduling method that finds a broadcast schedule of data for reducing the access time of the queries issued by mobile clients.

The rest of the paper is organized as follows. Section 2 shows the background of wireless data transmission, especially data retrieval on wireless channel. In Section 3 we shows the broadcast model and define the problem of data scheduling. After defining $QueryDistance(QD)$ measure, we analyze the complexity of the problem in this section. We illustrate the proposed method in Section 4 and evaluate the performance of the proposed method in Section 5. After the comparison with related works, we conclude this paper with some directions for further research.

2. Background

In this paper we assume that wireless computing environment consists of mobile clients and one fixed server system. (Our work is also applicable to the case of multiple server systems without loss of generality) Mobile clients do their work with or without the help of server system, i.e. the clients cooperate with server through one-way or two-way mode.

Two-way mode: The client sends requests to a server and receives server's replies.

One-way mode: The server repeatedly broadcasts data stream commonly called *bcast* to unspecified clients and the client accesses the data of interest in the broadcast stream.

The inherent restrictions of wireless systems, such as bandwidth limitation and energy restriction, recommend the mobile clients do their works in broadcasting mode as much as possible. The server system analyses the pattern of clients' requests and broadcasts data objects which are requested by lots of clients. The

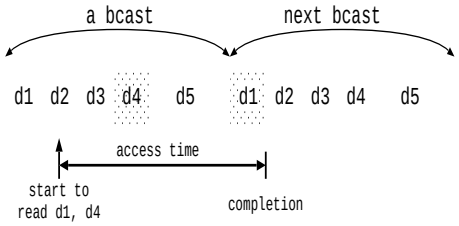

Figure 1. A Query onto the Wireless Broadcast

data objects on broadcasting channel can be accessed by clients without the need of request.

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* and a client issues a query retrieving d1 and d4. (In this work we assume there is no precedence relation in accessing data objects) 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 broadcast schedule is formed as < d2, d3, d1, d4, d5 >, then the client can retrieve d1, d4 in current *bcast*. This says that an efficient schedule provides access time reduction to mobile clients.

3. Problem Definition

We first explain some notations that will be used throughout the paper. d_i the a data object delivered by broadcasting, the size of which is denoted by $|d_i|$ (in packets). The set of data objects on broadcasting channel is $\mathcal{D}$ and is represented as $\mathcal{D} = \{d_1, d_2, \cdots, d_N\}$. B is the size of one broadcast stream (a bcast) and is computed as $B = \sum |d_i|$ for all $d_i \in \mathcal{D}$. The query q_i, which is issued by some mobile clients, accesses the data objects on the broadcasting channel and the data set accessed by a query q_i is denoted by $QDS(q_i)$. The set of queries is denoted by $\mathcal{Q} = \{q_1, q_2, \cdots, q_M\}$ and $freq(q_i)$ is the reference frequency of the query q_i. We use the notation σ for the broadcast schedule, denoted by $\sigma =< d_i, d_j, \cdots, d_k >$. The data objects are not replicated in a single *bcast*. $loc(d_i, \sigma)$ is the location of d_i in σ and $data(\sigma, i)$ is the data object in the i-th location in σ. Thus, in other words, $data(\sigma, i) = d_j$ is the same as $loc(d_j, \sigma) = i$. $DS(\sigma)$ is the set of data that is included in the given schedule σ.

The problem of scheduling wireless broadcast data is to find a broadcast schedule σ which minimizes 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 is dependent on the starting time of the query as shown in Figure 1. If, for example, the query is submitted at the beginning of current *bcast*, then both the data objects, d1 and d4, can be retrieved in one *bcast*. That is, the measure of $AT(q_i, \sigma)$ is not deterministic, which makes it difficult to develop data scheduling method for wireless broadcasting.

In this work we define a new measure named $QD(QueryDistance)$ for wireless broadcast data scheduling. This measure represents the coherence degree of a query's QDS and is defined as follows.

Definition 1 *Suppose $QDS(q_i)$ is $\{d^1, d^2, \cdots, d^n\}$, and δ^i is the interval between d^i and d^{i+1} in schedule σ, i.e. δ^i is $t^i - |d^i|$ where t^i is the duration from the start of d^i to the start of d^{i+1}. (For example, in Figure 1, the query accesses d_1 and d_4, thus there are two δ's, $|d_5|$ and $|d_2| + |d_3|$.) Then the QueryDistance(QD) of q_i on σ is defined as:*

$$QD(q_i, \sigma) = B - MAX(\delta^j)$$

□

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

$$\begin{aligned} &if \quad QD(q_i, \sigma_1) \geq QD(q_i, \sigma_2) \\ &\quad then \quad AT^{avg}(q_i, \sigma_1) \geq AT^{avg}(q_i, \sigma_2) \end{aligned}$$

Proof: *Omitted for space limitation. See the reference [3].* □

Let the total query distance denoted by $TQD(\sigma)$ be defined as $\sum_{q_i \in \mathcal{Q}} QD(q_i, \sigma) \times freq(q_i)$. Now we redefine the problem of scheduling wireless broadcast data using 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 scheduling problem is to find a broadcast schedule σ_i such that $TQD(\sigma_i)$ is minimum among all possible σ_i, $i = 1, \cdots$.* □

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

Proof: *Transformation from Optimal Linear Arrangement Problem [4]. See the reference [3].* □

Algorithm : QEM
Input: a set of data $\mathcal{D}$ and a set of queries $\mathcal{Q}$
Output: a broadcast schedule σ
Method:

1. Initially σ is empty.
2. Sort the queries in decreasing order $freq(q_i)$.
3. For each query q_i in the sorted order, expand σ with q_i by using the **QDS Expanding Rules.**

Figure 2. Algorithm Description

4. QDS Expanding Method (QEM)

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

4.1. Basic Idea

The proposed method constructs broadcast schedule by expanding 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 2 and the its basic policies are as follows.

Policy 1: Higher-frequency query takes precedence over the lower-one when expanding the schedule.

Policy 2: When expanding a query i.e., its QDS, the QD's of the queries which had been previously expanded remain unchanged.

Policy 3: When expanding the QDS of query q_i into currently constructed schedule, the proposed method always minimizes the QD of q_i as much as possible.

Now we take a simple example and show how QEM constructs broadcast schedule. Let us assume that there are 8 data objects to be broadcasted and 3 queries that mobile clients submit onto the broadcasting channel. The data set of each query (i.e., QDS) is depicted as in Figure 3. All data objects are assumed to be of equal size and the occurrence frequency of each query is assumed as $freq(q_1) = 3, freq(q_2) = 2$ and $freq(q_3) = 1$.

Initially the schedule σ^{step0} is empty. According to *Policy* 1, the algorithm finds the highest frequency

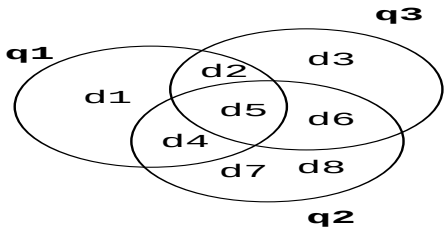

Figure 3. Queries and their QDS's

query, q_1, and expands its QDS. Then current schedule σ^{step1} is formed as this:

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

The data objects in the above are interchangeable one another, for $QD(\sigma^{step1}, q_1)$ is not varied. We use the symbols '[' and ']' between data objects that can be interchangeable without changing the TQD value. For notational convenience, we will not use angle brackets ('<' and '>') for intermediate results, if there is no ambiguity. That is, we use angle brackets for only final schedules.

Second the algorithm expands the 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 is expanded into one of these 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 the data objects bounded by '[' and ']' 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 (*Policy* 3) while preserving the QD of q_1 unchanged i.e., $QD(q_1, \sigma^{step1}) = QD(q_1, \sigma^{step2})$ (*Policy* 2).

Finally the $QDS(q_3)$ is expanded. Among the data objects in $QDS(q_3)$, only d_3 is not included in the current schedule σ^{step2}. To insert it into the schedule increases the QD of q_1 and(or) q_2, which violates *Policy* 2 of our method. Hence d_3 has to be appended to the left or right end of σ^{step2} . When appending d_3, the data objects of d_2, d_5, and d_6 have 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 the 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 will be one of the forms below which are results of $\sigma^{step3}_{LeftAppend}$ by possible interchanging data objects.

$$\begin{aligned} & < d_3,\ d_1,\ d_2,\ d_4,\ d_5,\ d_6,\ d_7,\ d_8 > \\ or\ & < d_3,\ d_2,\ d_1,\ d_4,\ d_5,\ d_6,\ d_7,\ d_8 > \\ or\ & < d_3,\ d_1,\ d_2,\ d_5,\ d_4,\ d_6,\ d_7,\ d_8 > \\ or\ & < d_3,\ d_2,\ d_1,\ d_4,\ d_5,\ d_6,\ d_8,\ d_7 > \\ & and\ \ so\ \ on. \end{aligned}$$

4.2. QEM Description

After describing a few lemmas and a definition we explain the proposed scheduling method.

Lemma 2 *If a query q_i accesses d_i and d_j on the schedules, σ_1 and σ_2 in the below, then then $QD(q_i, \sigma_1) = QD(q_i, \sigma_2)$.*

$$\begin{aligned} \sigma_1 &= < \cdots, d_{i-1}, d_i, d_{i+1}, \cdots, d_{j-1}, d_j, d_{j+1}, \cdots > \\ \sigma_2 &= < \cdots, d_{i-1}, d_j, d_{i+1}, \cdots, d_{j-1}, d_i, d_{j+1}, \cdots > \end{aligned}$$

□

This lemma is a direct consequence from the definition of *QueryDistance*. The lemma 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 $\sigma_1 \equiv \sigma_2$* □

Lemma 3 *If a schedule σ_2 is an inverse of σ_1, as in the below, then $\sigma_1 \equiv \sigma_2$.*

$$\begin{aligned} \sigma_1 &= < d_1, d_2, \cdots, d_{N-1}, d_N > \\ \sigma_2 &= < d_N, d_{N-1}, \cdots, d_2, d_1 > \end{aligned}$$

Proof: *For all j, δ_j in σ_1 is equal to that in σ_2. So $QD(q_i, \sigma_1)$ is equal to $QD(q_i, \sigma_2)$ for all q_i.* □

In the algorithm the data broadcast schedule is composed of a few constructs, described in the below, based on EBNF (Extended Backus-Naur Form) methodology. This constructs are internally used in our method and not parts of real broadcast schedule.

schedule ::= segment < '⊕' segment >
segment ::= fragment < fragment >
fragment ::= '[' component ']'
component ::= element < ',' element >
element ::= data | '[' component ']'

A schedule consists of segments(s_i) which are connected by '⊕' symbol. The ordering of segments are flexible within a schedule. If, for example, a schedule σ is formed as "$s_1 \oplus s_2 \oplus s_3$", then the schedules "$s_1 \oplus s_3 \oplus s_2$", $\cdots$, "$s_2 \oplus s_1 \oplus s_3$" 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 kinds of constituents bounded by '[' and ']' can be freely interchanged with others within the boundary, which is based on Lemma 2.

Example 1

$$\begin{aligned} \sigma &= \overbrace{[d_1, d_2][d_3][d_4, d_5, [d_6, d_7], d_8]}^{s_1} \oplus \overbrace{[d_9]}^{s_2} \oplus \overbrace{[d_{10}]}^{s_3} \\ &\equiv [d_2, d_1][d_3][d_5, d_8, [d_7, d_6], d_4] \oplus [d_{10}] \oplus [d_9] \\ &\equiv [d_9] \oplus [d_{10}] \oplus [d_2, d_1][d_3][[d_7, d_6], d_4, d_5, d_8] \\ &\equiv \ and\ \ so\ \ on. \\ \sigma' &= \underbrace{[d_3][d_1, d_2]}[d_4, d_5, [d_6, d_7], d_8] \oplus [d_9] \oplus [d_{10}] \end{aligned}$$

The schedule σ consists of 3 segments. s_1 has 3 fragments and s_2 and s_3 has one fragment each. 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. □

Our method constructs broadcast schedule by inserting the QDS of each query based on greedy philosophy, that is the QD of higher frequency query is guaranteed not to be prolonged when minimizing the 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 query's data set.

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

in current schedule. By the operation MDC the data objects, which are included in the QDS and the given schedule, are moved closest for the purpose of minimizing the QD of the query.

Definition 4

MRF$(f_i, QDS(q_k))$**:**

1. *In the given fragment f_i find the elements e_i such that $DS(e_i) \subset QDS(q_k)$. Then remove them from f_i and make them as a new fragment f_γ. f_i becomes f_i' after the deletion.*

2. *In f_i' find the elements e_j such that $DS(e_j) \cap QDS(q_k) \neq \emptyset$. 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'.*

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

3. *When assuming that f_i'' is f_i' from which the elements e_j have been removed, the result of $MRF(f_i, QDS(q_k))$ has the form of :*

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

MRS$(s_i, QDS(q_k))$**:**

1. *Among the fragments in s_i, find the left-most fragment f_i such that $DS(f_i) \cap QDS(q_k) \neq \emptyset$.*

2. *Order the fragments of s_i like this:*

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

MDC$(s_i, QDS(q_k))$**:**

1. *Find the left-most and right-most fragments (respectively f_L and f_R) in s_i such that $DS(f_L) \cap QDS(q_k) \neq \emptyset$ and $DS(f_R) \cap QDS(q_k) \neq \emptyset$.*

2. *If $f_L \neq f_R$, then order the fragments of s_i as follows.*

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

3. *Otherwise, s_i is*
$\cdots f_{L-1} MLF(f_L, DS(f_L) \cap QDS(q_k)) f_{R+1} \cdots$, *or*
$\cdots f_{L-1} MRF(f_L, DS(f_L) \cap QDS(q_k)) f_{R+1} \cdots$

In case of $MLS(s_i, QDS(q_k))$, the fragments are arranged like "$\cdots \ f_{i-1} \ MLF(f_i, DS(f_i) \cap QDS(q_k)) \ f_{i+1} \ \cdots$" where f_i is the right-most fragment in s_i such that $DS(f_i) \cap QDS(q_k) \neq \emptyset$. And, in case of $MLF(f_i, QDS(q_k))$, the fragments are arranged like "$f_\gamma \ f_\beta \ MLF(f_\alpha, DS(f_\alpha) \cap QDS(q_k)) \ f_i''$". □

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

$f_i = [d_1, d_2, [d_3, d_4], d_5, [d_6, [d_7, d_8], d_9] \ ]$,
$s_i = [d_1, d_2] \ [d_3, d_4] \ [d_5, [d_6, [d_7, d_8], d_9] \ ]$,
$QDS(q_1) = \{d_2, d_4, d_6, d_8, d_{10}\}$,
$QDS(q_2) = \{d_2, d_4, d_6, d_8\}$.

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

$MRF(f_i, QDS(q_1)) = [d_1, d_5][d_9][d_7][d_8][d_6][d_3, d_4][d_2]$,
$MRS(s_i, QDS(q_1)) = [d_1][d_2][d_3, d_4][d_5, [d_6, [d_7, d_8], d_9]]$,
$MDC(s_i, QDS(q_2)) = [d_1][d_2][d_3, d_4][d_6][d_8][d_7][d_9][d_5]$.

□

Lemma 4 *The basic operations (in Definition 4) preserve the QD's of the queries that have been previously used for constructing the given 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 expanding the schedule with a new $QDS(q_k)$. 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 $MRF(f_i, QDS(q_1))$, $MRS(f_i, QDS(q_1))$ and $MDC(s_i, QDS(q_2))$.* □

QDS Expanding Rules: When expanding the 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. if $DS(s_i) \cap QDS(q_i) = \emptyset$ for all s_i, then make a new segment s_{k+1} with the data in $QDS(q_i)$ and set $\sigma' = \sigma \oplus s_{k+1}$.

2. if there is one segment s_i such that $DS(s_i) \cap QDS(q_i) \neq \emptyset$, then do the followings.

   (a) if $QDS(q_i)$ is included in $DS(s_i)$, then $\sigma' = \cdots \oplus s_{i-1} \oplus s_i' \oplus s_{i+1} \oplus \cdots$, where 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 following form:

      i. when s_i consists of one fragment:

$$[\ s_i \ , \ QDS(q_i) - DS(s_i) \ ]$$

ii. when s_i consists of more than one fragment[1]:

$$s_i\ [\ QDS(q_i) - DS(s_i)\ ]$$

(c) otherwise, transform s_i as one of these 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. if the data of $QDS(q_i)$ are spread over two segments s_i and s_j, then do as follows:

(a) if all the data objects in both segments are included in $QDS(q_i)$, then two segments are transformed into one segment with this form:

i. when s_i and s_j consist of one fragment:

$$[s_i, s_j, QDS(q_i) - (DS(s_i) \cup DS(s_j))]$$

ii. when only s_i consists of one fragment:

$$s_j[s_i, QDS(q_i) - (DS(s_i) \cup DS(s_j))]$$

iii. when only s_j consists of one fragment:

$$s_i[s_j, QDS(q_i) - (DS(s_i) \cup DS(s_j))]$$

iv. when s_i and s_j consist of more than one fragment:

$$s_i\ s_j[QDS(q_i) - (DS(s_i) \cup DS(s_j))]$$

(b) if all the data objects in one segment s_i is included in $QDS(q_i)$, then combine s_i and s_j and make a new segment with following form:

i. when s_i consists of one fragment:
$[s_i, QDS(q_i) - (DS(s_i) \cup DS(s_j))]$
$MLS(s_j, QDS(q_i))$

ii. when s_i consists of more than one fragment:
$s_i, [QDS(q_i) - (DS(s_i) \cup DS(s_j))]$
$MLS(s_j, QDS(q_i))$

(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 whose $QD(q_i, \sigma_x)$ is not larger than the others among the followings.

$\sigma_{LeftAppend} = [QDS(q_i)']s_iMLS(s_j, QDS(q_i))$

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

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

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

[1] "$s_j[some\ schedule]$" is short for "$f_1 f_2 \cdots f_k[some\ schedule]$" where f_i (i=1,...,k) is a fragment of s_j.

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

(a) 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 $QDS(q_i) \cap DS(s_j^d) = \emptyset$

(b) 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 $d_j \in QDS(q_i)$.

(c) Make the schedule as follows.

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

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

5. Experiments and Results

In this section we test our method with several environmental parameters which are described below. We measure the performance improvement i.e., TAT reduction, of the proposed method against the random schedules.

Number of data objects (N): The number of data objects which are delivered on broadcasting channel. Every data object on broadcast channel is accessed by one or more queries.

Number of queries (M): The number of queries which access parts of the broadcast data set. Every query accesses at least one data object on broadcasting channel and the QDS's are mutually independent between queries.

Selectivity: 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.

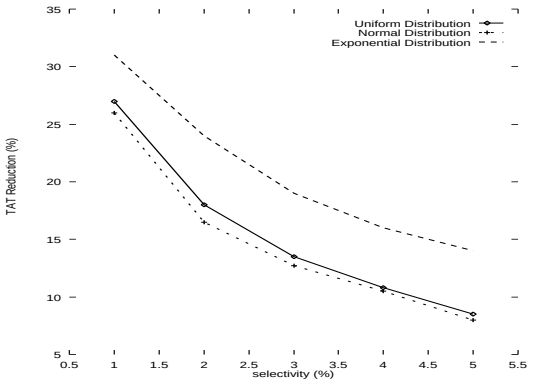

Figure 4. TAT reduction with change in the selectivity

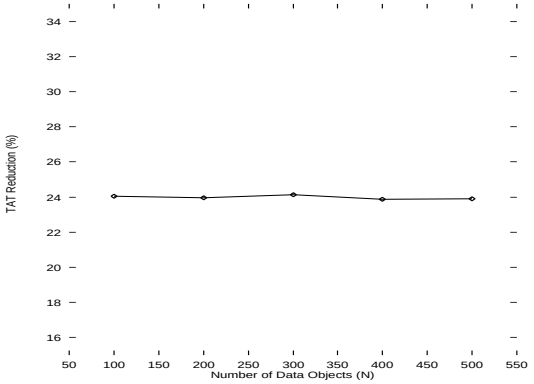

Figure 5. TAT reduction with change in the number of data objects

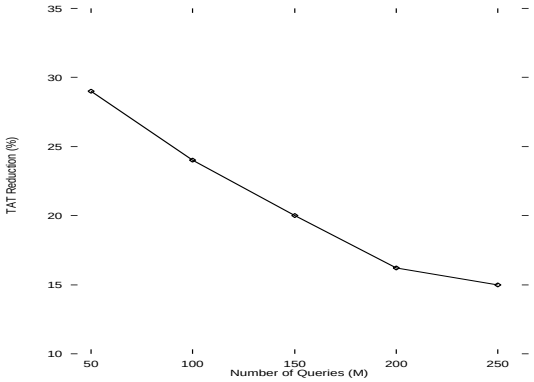

Figure 6. TAT reduction with change in the number of queries

Distribution of $freq(q_i)$: We consider 3 different kinds of distribution of query's occurrence frequency: uniform distribution, normal distribution, and exponential distribution (λ=1). In all the three distributions we set the minimum frequency into 1 and maximum one into M.

Size of data object: We fix the size of one data object to be 10 packet lengths. A packet is considered the basic unit of data manipulation on wireless channel. We have experimented with several other values, but the results are almost equal. We assume the size of each data object to be equal.

In the first experiment we change selectivity values with 1000 data objects and 100 queries. The result of the experiment is illustrated in Figure 4. The performance improvement goes up 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. With this we can argue that the proposed method performs better when the query's reference distribution is skewed. This is because our method adopts the greedy philosophy, using it tries to reduce the QD of higher-frequency query. 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 5. As shown in this result, the performance of QEM has little dependency on the number of data objects.

In the third experiment we change the number of queries while fixing N (1000) and selectivity (2%). The result in Figure 6 shows the improvement of QEM decreases with large number of queries.

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

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

Figure 4 and Figure 6 are based on the same $\mathcal{DOD}$ variation, in spite of different M's and selectivities, so show almost similar improvement pattern. In other words the performance of QEM is dependent on $\mathcal{DOD}$ essentially.

However, when comparing Figure 4 with Figure 6, 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 4) gives better result than that of 50 queries with 2% (in Figure 6) regardless of the same 100% $\mathcal{DOD}$.

6. Related Work

There are some researches on wireless data broadcasting. The indexing on wireless channel [5, 6] reduces the tuning time by indicating mobile clients to skip data and(or) index portion of no interest. The works [1], [2], [8] and [7] reduce clients' access time by efficient organization of the broadcast data stream.

The proposed method QEM reduces the access time by efficient scheduling wireless broadcast data. What follows are the features of the previous work which are different from our method.

BroadcastDisks[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. But this way of broadcasting increases the *bcast* length, so it gives longer average access time to the users who access less popular data objects. Our approach provides no such penalties by keeping equal delivery frequencies of all data objects, for ours only controls the delivery order of broadcast data. In addition *BroadcastDisks* approach does not consider that a query accesses more than one data object.

The scheduling method in [7] makes the broadcast schedule by using stochastic model. But it considers only the access frequency and delivery interval of each data object. That is, the approach does not consider the relationship between data objects although a query can access more than one data objects. And this approach has similar shortcommings of *Broadcast Disks* approach in the above by prolonging the broadcast cycle.

We considers a environment where a query can access more than one data object. But no previous work considers such environment to the best of our knowledge. Our work is different from the previous works primarily in this point.

7. Summary

In this paper we have introduced the problem of wireless broadcast data scheduling based on the assumption that the user queries access more than one data object on wireless broadcast channel and each query has different occurrence frequency. Through the address of the problem, we show it has much effect on access time how to schedule the broadcast data.

We have analyzed the problem NP-complete by using the proposed QD measure and also proposed a method QEM. By the use of some examples we have explained the method in detail.

The performance of the proposed method is experimented with several environmental parameters. Based on the result we observe QEM effectively construct wireless broadcast schedule and give about 20% of access time reduction in case of about 200% $\mathcal{DOD}$.

Compared with the related works, our approach considers the environment where a user query accesses more than one data object and it does not prolong the total broadcast cycle.

As a further work, we will explore the broadcast data scheduling method in multi-channel environment and non-uniform broadcasting environment.

Acknowledgements

The authors would like to thank the reviewers for their helpful comments on the paper.

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*, pages 199–210, 1995.

[2] S. Acharya, M. Franklin, and S. Zdonik. "Disseminating Updates on Broadcast Disks". In *Proceedings of Very Large Data Bases Conference*, pages 354–365, 1996.

[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] M. R. Garey and D. S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness.* Freeman Publishing Company, 1976.

[5] T. Imielinski, S. Viswanathan, and B. R. Badrinath. "Data on Air : Organization and Access". Technical report, Rutgers University, 1994.

[6] T. Imielinski, S. Viswanathan, and B. R. Badrinath. "Energy Efficient Indexing On Air". In *Proceedings of ACM SIGMOD Conference*, pages 25–36, 1994.

[7] C. Su, L. Tassiulas, and V. J. Tsotras. "Broadcast Scheduling for Information Distribution". *Wireless Networks*, 1998.

[8] K. Tan and J. X. Yu. "Generating Broadcast Programs that Support Range Queries". *IEEE Transactions on Knowledge and Data Engineering*, 10(4), 1998.