Data Scheduling in Multiple Channel/Multiple Receiver Environments for Processing Data Requests with Timing Constraints

GUANLING LEE, YI-NING PAN AND ARBEE L. P. CHEN*
Department of Computer Science and Information Engineering
National Dong Hwa University
Hualien, 974 Taiwan
E-mail: guanling@mail.ndhu.edu.tw
**Department of Computer Science*
National Chengchi University
Taipei, 116 Taiwan

In real-time environments, information is disseminated to clients under timing constraints. In this paper, we focus on the real time data scheduling problem in multiple broadcast channels environments, where the clients are equipped with multiple receivers. Each request is associated with a deadline. The clients can either retrieve data items from the broadcast channels or send requests to the server and then listen to the broadcast channels. The aim of our work is to serve as many requests as possible.

In our approach, the content of the broadcast program is first decided. Afterwards, the proposed periodic broadcast program generation algorithm and the on-demand broadcast program generation algorithm are employed. Simulations were performed to show the benefit of our approach. We conclude that our data scheduling algorithms are scalable, and that the time spent on executing our data scheduling algorithms is low. Moreover, when our data scheduling algorithms are used, the percentage of requests that miss their deadlines is also low.

Keywords: real-time database, timing constraint, periodic broadcast program, on_demand broadcast program, multiple broadcast channels

1. INTRODUCTION

Rapid advances in wireless communications and software/hardware technologies now enable a client carrying a mobile device to access information without time or location restrictions. Broadcast-based information systems disseminate information with a cost that is independent of the number of clients, which compensates for limited bandwidth in wireless environments.

In broadcast data delivery environments, there are two basic modes of data transmission. In the first mode, the *pull mode*, clients request data items from the server via an *uplink channel*. The server determines which request will be satisfied next and then disseminates the required data item via a *broadcast channel*. In [4], many traditional pull-based scheduling strategies were compared. In the *First Come First Served* scheduling strategy, requests are served in the order of their arrival times. However, popular data

Received September 30, 2003; revised September 17, 2004; accepted November 23, 2004.
Communicated by Suh-Yin Lee.

items are requested more frequently, and the same data item may be disseminated to clients repetitively, which leads wasted of bandwidth. In the *Most Request First* scheduling strategy, the data item with the maximum number of requests is selected for broadcast. Therefore, requests for unpopular data items may have very long waiting times or may never be served. To avoid the starvation situation, the *Longest Wait First* scheduling strategy was proposed. In [4], the authors proposed a promising scheduling strategy, called RxW, which considers not only the number of requests for each object but also the amount of waiting time of the oldest requests for the data item. The effectiveness of the batch scheme for some pull-based scheduling strategies was also been examined in [22].

In the second mode, the *push mode*, the server transmits a pre-selected data item set on broadcast channels according to a *broadcast program*. The broadcast program determines the order of data items in the broadcast channels. The clients can retrieve broadcast data by tuning in to the broadcast channels. The simplest method for organizing broadcast data items is *flat* organization [16]. In flat organization, the server broadcasts the combination of data items needed by clients periodically. However, the access frequency of each data item is not the same in practice. Therefore, in [2, 3, 15], the *non-flat* organization method, which broadcasts frequently accessed data items more often than unpopular ones, was proposed.

In recent years, many researches have studied of multiple-input/multiple output wireless systems, because they make use of the spatial dimension of a channel to provide considerable capacity and to increase resistance to fading [1, 13, 23, 26]. In [21], the problem of generating broadcast program on multiple channels was studied. In that approach, data items together with an index of them form an index tree. The index nodes on each level of the index tree are assigned to an individual channel. The client tunes in to the first channel to get the root of the index tree and then follows the index pointers to retrieve the desired data item. Therefore, the number of channels needed to allocate the index nodes is determined by the depth of the index tree, which is not flexible. Allocating each index node in an individual channel wastes space. An algorithm that avoids these drawbacks was proposed in [18]. This algorithm tries to find the optimal allocation of the index and data items so as to minimize the average access latency for any number of broadcast channels. In [12], a data scheduling algorithm which considers index and data item replication in the multiple broadcast channel environment was proposed. In [15], a log-time algorithm for distributing broadcast data items over multiple broadcast channels was proposed. This algorithm was modified for transmitting data items with errors. A near optimal algorithm for generating a broadcast program over multiple broadcast channels without am indexing structure was proposed in [10].

There is an integrated mode, called the *hybrid mode*, which combines the push and pull modes together. In [5], data items were classified as broadcast or on-demand modes. The server multiplexes the push and pull data for dissemination. Therefore, clients can either listen to the broadcast channels to retrieve the desired data items or submit requests via the uplink channel and then wait for the data items to be transmitted in the on-demand broadcast channel.

Previous researches on broadcast scheduling assumed that once a client generates a request, this request will not be discarded until it is satisfied. However, in real applications, a request may be associated with a deadline. For example, if a client needs traffic information to decide which road to take, he must receive the response before he reaches the crossroad. Some recent works have begun to address the transmission of a real-time database. The *pinwheel scheduling problem*, which tries to schedule a real-time database for satellite-based communication, was first introduced in [14]. This problem was generalized in [8] and applied to generate fault-tolerant, real-time broadcast disks in [6]. Since impatient users may withdraw their requests before they are served, a data broadcast scheduling algorithm for such users was proposed in [17]. Moreover, an adaptive hybrid transmission technique based on the PinOpt [8] algorithm was presented in [24] and [11]. In this model, information is disseminated to clients with timing constraints. Based on [11], we discussed the problem of transmission data items with timing constraints in a multiple broadcast channel environment [20]. However, the assumption that the number of receivers should be equal to the number of channels and the lack of a performance study were two main drawbacks.

In this paper, by extending [20], we study the problem of transmitting data items with timing constraints in m broadcast channels in which the clients are equipped with r receivers. In our approach, the data items are divided into two sets, a *broadcast data set* and an *on-demand data set.* The data items in the broadcast data set are periodically broadcast on the broadcast channels, while the data items in the on-demand data set are transmitted when they are requested. The server offline allocates the data items in the broadcast data set to multiple broadcast channels. The clients that retrieve data items from the broadcast channels are guaranteed to receive them within the time limit. Once the desired data item is no longer included in the broadcast channels, clients can send requests associated with deadlines to the server. By using the bandwidth remaining for the on-demand mode, the server broadcasts the requested data items in an online fashion.

The rest of this paper is organized as follows. In section 2, the scheduling problem is formulated and the system architecture is introduced. The scheduling algorithms for the broadcast data set and the on-demand data set are presented in section 3. In section 4, a simulation model and analysis of the simulation results are given. Finally, section 5 concludes this work.

2. WIRELESS DATA DELIVERY MODEL

In this section, the real-time scheduling problem and the system architecture are introduced.

2.1 Problem Formulation

A database in a server consists of many data items. The data items may have various sizes and are conceptually split into several *pages*. Moreover, to enable clients to have the ability to reorder data items, an additional piece of information, i.e., a page number, is added to each page. Assume the time needed to broadcast a single data page is called one *time slot*. Each data item X is characterized by $X(X.s, X.p)$. $X.s$ denotes the *size* of data item X, which is equal to the number of pages of data item X. $X.p$ denotes the *period* of data item X, which is the timing constraint of data item X. That is, the number of time slots needed to retrieve data item X can not exceed $X.p$. Therefore, data item X in the broadcast data set will be broadcast $X.s$ pages every $X.p$ consecutive time slots to ensure that a client that needs data item *X* in the broadcast data set is guaranteed to receive all the pages of data item *X* within a period of time not greater than *X.p*.

In the on-demand mode, a client sends a request associated with a deadline to the server and then listens to the broadcast channels until the desired data item is broadcast or until an acceptable waiting time is exceeded.

The performance metric of our problem is not a minimal the average access time but the percentage of information demands satisfied by the server.

2.2 System Architecture

Server Side: The broadcast server is divided into two modes, the broadcast mode and the on-demand mode. Each data item in the database is classified as belonging to one of these two modes, denoted as a *broadcast data set* and an *on-demand data set*, respectively. Moreover, the bandwidth for broadcast is divided into $m + 1$ channels. One of the channels is reserved for the delivery of *broadcast schema*. The broadcast schema contains information about the broadcast data set. The broadcast scheduler generates a periodic broadcast program on the m broadcast channels. Because the clients are equipped with multiple receivers, the pages of a data item can appear in more than one channel at the same time. By using the bandwidth remaining for the on-demand mode, the on-demand scheduler selects a request to be served and broadcasts the requested data items in an online fashion.

Client Side: Each client is equipped with *r* receivers. The clients can require one data item per request, and each request is associated with a deadline. When a client needs a data item, it first tunes into the broadcast channels to retrieve the broadcast schema. By examining the broadcast schema, the client can determine whether he can get the data item from the broadcast channels. If the needed data item is in the broadcast data set, the client tunes into the broadcast channels and retrieves the desired data item. Otherwise, the client sends a request to the server via the uplink channel and listens to the broadcast channels to retrieve the data pages. Since the time when each client tunes into the broadcast channels is unpredictable, a client may receive page 5 first, and then page 1, page 2, page 3, and page 4. By using the information, that is, the page number, stored in each page, the clients can reorder the data pages themselves.

3. DATA SCHEDULING

3.1 Preliminary

In this subsection, the generalized pinwheel task scheduling problem [9], which we employee to generate our broadcast program, is introduced.

Generalized Pinwheel Task Problem: Given a multiset $\{(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)\}$ of ordered pairs of positive integers, determine an infinite sequence over the symbols $\{1, 2, 3, \ldots, n\}$ such that, for each i, $1 \leqq i \leqq n$, any subsequence of b_i consecutive symbols contains at least a_i i's.

Each generalized pinwheel task X can be described by an ordered pair of positive integers ($X.a$, $X.b$), where $X.a$ denotes the computation requirement and $X.b$ denotes the window size. The *weight* of a task X is $\frac{X.a+1}{X.b}$ and is denoted by $X.w$. Given a feasible task set, the Pinfair algorithm [9] can construct a pinwheel schedule in single resource environments; that is, for each task X in the feasible task set, it is expected to allocate the shared resources for at least $X.a$ times out of every $X.b$ consecutive time slots.

We observe that the generalized pinwheel scheduling problem is similar to the *single channel with single receiver* (SCSR) broadcast scheduling problem. Each data item X can be viewed as a task X, where $X.s$ and $X.p$ are mapped to $X.a$ and $X.b$, respectively. Moreover, a single broadcast channel is mapped to a single shared resource. By transforming the generalized pinwheel scheduling problem into the broadcast scheduling problem, we can apply the Pinfair algorithm to solve the SCSR broadcast scheduling problem; that is, if data item X is broadcast in a broadcast channel, it is expected to be broadcast for at least $X.s$ pages out of every $X.p$ consecutive time slots.

The definitions of the terms which will be used to introduce the Pinfair algorithm are as follows:

- allocated(X, t) = the number of resources allocated to a task X during the interval (0, t).
- earliest(X, j) = the earliest time when a task X can be scheduled at the jth times, defined by $\left\lfloor \frac{j}{X.w} \right\rfloor$.
- latest(X, j) = the latest time when a task X must be scheduled at the jth times, defined by $\left\lfloor \frac{j+1}{X.w} \right\rfloor - 1$.
- A task X is said to be *contending* if allocated (X, t) = k and earliest (X, k) $\leqq t$. The *pseudo-deadline* $X.d$ is defined as the *latest* (X, k).

Two problems were encountered when we developed the scheduling algorithm [7]. One is the *decision problem*, and the other is the *scheduling problem*. The decision problem is to determine whether a given instance is feasible. The scheduling problem is to actually construct a schedule for a given feasible instance. The following theorem, proposed and proved in [9], solves the decision problem, while the Pinfair algorithm is the solution of the other problem.

Theorem 1 Any system of generalized pinwheel tasks Γ satisfying $\sum_{X \in \Gamma} X.w \leq 1$, where $X.w$ is the weight of a task X and is defined as $\frac{X.s+1}{X.p}$, can be successfully scheduled by Algorithm Pinfair.

The *Pinfair algorithm* is described below:

Algorithm Pinfair (Γ), where Γ is a system of pinwheel tasks.

Step 1: For each task $X \in \Gamma$, define a weight $X.w$ as $\frac{X.s+1}{X.p}$.

Step 2: If $\sum_{X \in \Gamma} X.w > 1,$ return failure.

Step 3: Allocate the resource at each time slot to the contending task with the tightest pseudo-deadline.

In our approach, the problem of transmitting data items with timing constraints in multiple channel environments where clients are equipped with multiple receivers is considered. We formulate our problem as the *Multiple Channels with Multiple Receivers* (*MCMR*) broadcast scheduling problem and discuss it in the following subsection.

3.2 Periodical Broadcast Program Generation

To solve the MCMR broadcast scheduling problem, we first transform it into the SCSR broadcast schedule problem. By solving SCSR broadcast schedule problem, we can solve the MCMR broadcast schedule problem. In subsection 3.2.1, we explain how the MCMR broadcast schedule problem can be transformed into the SCSR one. How the broadcast data set can be determined discussed in subsection 3.2.2. The periodical broadcast program generation algorithm is presented in subsection 3.2.3.

3.2.1 Problem transformation

In our approach, the MCMR broadcast scheduling problem is first transformed into the SCSR one. In this transformation, the time slots C_{nm} in the multiple broadcast channels are mapped to consecutive time slots in the single broadcast channel. The mapping sequence is C_{01} C_{02} … C_{0m} C_{11} C_{12} … C_{nm}, where n denotes the number of time slots and m denotes the number of broadcast channels. Fig. 1 shows a graph of the mapping sequence. Moreover, each data item $X(X.s, X.p)$ in the MCMR environment is mapped to $X(X.s, X.p * m)$ in the SCSR environment. After the broadcast program is constructed, the time slots are mapped to their original positions.

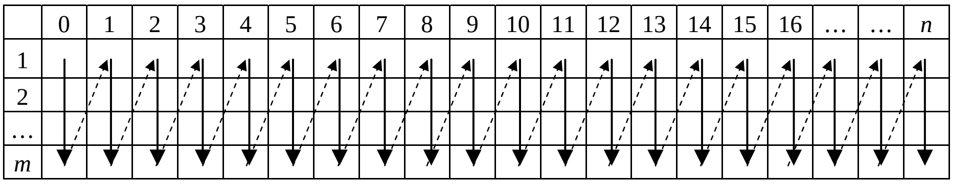


Fig. 1. Graphical representation of the mapping sequence.

3.2.2 Determine the broadcast data set

When the MCMR broadcast scheduling problem is transformed into the SCSR broadcast scheduling problem, the data item $X(X.s, X.p)$ in the MCMR environment is mapped to $X(X.s, X.p * m)$ in the SCSR environment. To satisfy Theorem 1, the *weight* of data item X is defined as $\frac{X.s+1}{X.p*\min(m,r)}$, where m is the number of channels, r is the number of receivers, and min(m, r) returns the smaller value between {m, r}. By extending Theorem 2 proposed in [9] and considering the relationship between m and r, we get Lemma 1.

Lemma 1 The weight of data item X mapped from the MCMR scheduling problem to the SCSR scheduling problem is defined as $X.w = \frac{X.s+1}{X.p * \min(m, r)}$.

Proof: In the worst case of the scheduling algorithm, data item X is scheduled in the earliest time slot in the jth allocation, and for the next allocation, the (j + 1)th allocation of data item X, it is scheduled in the latest time slot. A client is guaranteed to receive at least $X.s$ pages for every $X.p * m$ consecutive slots. Moreover, due to the constraint imposed on the number of receivers, the number of allocation times of data item X within m consecutive slots can not exceed r. Therefore, the interval of the latest(X, $j + X.s$) – earliest(X, j) can never exceed $X.p$ * min(m, r), that is,

$$\begin{aligned}
\text{latest}(X, j+X.s) - \text{earliest}(X, j) &= \left\lceil \frac{j+X.s+1}{X.w} \right\rceil - 1 - \left\lfloor \frac{j}{X.w} \right\rfloor \\
&= \left\lceil \frac{j}{X.w} + \frac{X.s+1}{X.w} \right\rceil - \left\lfloor \frac{j}{X.w} \right\rfloor - 1 \\
&= \left\lceil \frac{j}{X.w} \right\rceil + \frac{X.s+1}{(\frac{X.s+1}{X.p*m})} - \left\lfloor \frac{j}{X.w} \right\rfloor - 1 \\
&\leq X.p * \min(m, r) + 1 - 1 \\
&= X.p * \min(m, r).
\end{aligned}$$

❑

After defining the weight of each data item, we will address the problem of selecting the optimal broadcast data set. The broadcast data set is optimal if the following two conditions hold:

1. $\sum_{X \in \text{Broadcast Data Set}} X.w \leq \max(1, m/r)$ (according to Theorem 1)
2. $\sum_{X \in \text{Broadcast Data Set}}$ access frequency of data item X is the maximum.

Solving this problem is the same as solving the 0-1 knapsack problem. Although the 0-1 knapsack problem is NP-complete, many programming skills, such as dynamic programming, can be used to solve it [19, 25]. The details of the algorithm are omitted here. After we compute the optimal broadcast data set, the remaining data items in the database server not included in the broadcast data set are said to be in the on-demand data set. Following is a description of the *Broadcast Data Set Selection Algorithm*:

Algorithm Broadcast Data Set Selection (Γ)
Step 1: (The Pruning Step)
For any data item *X* in the previous broadcast data set, if the access frequency of data item *X* is less than $\frac{\text{previous broadcast cycle length}}{X.p}$, then do not assign data item *X* to the broadcast data set.
Step 2: Solve the 0-1 knapsack problem.
Step 3: Assign the remaining data items to the on-demand data set.

For each data item *X*, we check in the pruning step whether it is worth broadcasting. The purpose of the pruning step is to prune those data items whose numbers of requests are lower than the numbers of times they were broadcast in the previous broadcast cycle. Moreover, the *broadcast cycle length* is determined by the least common multiple of all the periods of the data items in the broadcast data set.

3.2.3 The periodical broadcast program generation algorithm

The steps in the *Broadcast Program Generation Algorithm* are as follows:

Algorithm Broadcast Program Generation
Step 1: Map each time slot in the MCMR environment to the SCSR environment.
Step 2: Select an optimal broadcast data set.
Step 3: For each time *T*, allocate the time slot to the contending data item whose allocated times within the m consecutive slots is smaller than *r* in the broadcast data set with the tightest pseudo-deadline.
Step 4: Map each time slot in the SCSR environment to the MCMR environment.
Step 5: For each time *T*, record the number of unallocated time slots in *num*(*T*).

For instance, assume that three channels are available, and that each client is equipped with 3 receivers. Consider an optimal broadcast data set that contains *A*, *B*, *C*, and *D*, each of which is characterized as {*A*(21, 20), *B*(4, 20), *C*(9, 10), *D*(1, 40)}. Thus, the broadcast cycle length is 40. After transformation, the broadcast data set is {*A*(21, 60), *B*(4, 60), *C*(9, 30), *D*(1, 120)}, and the broadcast cycle length is 120. In the following, we discuss the process of generating the broadcast program in the SCSR environment.

Slot	0	1	2	3	4
Data Item	A_1	C_1	A_2	C_2	B_1

k = allocated(*A*, *t*)	1	1	2	2	2
(earliest(*A*, *k*), latest(*A*, *k*))	(0, 2)		(2, 5)		
k = allocated(*B*, *t*)	0	0	0	0	1
(earliest(*B*, *k*), latest(*B*, *k*))	(0, 11)	(0, 11)	(0, 1)	(0, 11)	(0, 11)

k = allocated(C, t)	0	1	1	2	2
(earliest(C, k), latest(C, k))	(0, 2)	(0, 2)		(3, 5)	
k = allocated(D, t)	0	0	0	0	0
(earliest(D, k), latest(D, k))	(0, 59)	(0, 59)	(0, 59)	(0, 59)	(0, 59)

In $T = 0$, all the data items are in the contending mode; thus, we allocate the 0th time slot to the contending data item A with the smallest pseudo-deadline, 2. In $T = 1$, we observe that only data items B, C, and D are in the contending mode. Comparing the pseudo-deadlines of these three data items, we allocate the 1st time slot to data item C. As we proceed, a broadcast program with a length of 120 is generated. Mapping the resultant broadcast program from the SCSR environment to the MCMR environment, we get the following broadcast program with a length of 40:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	A_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	A_{11}	A_{12}	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_1	A_1
2	C_1	B_1	D_1	A_5	A_6	A_7	A_8	A_9	A_{10}	C_1	C_2	A_{14}	B_4	A_{16}	A_{17}	A_{18}	A_{19}	A_{20}	A_{21}	C_2
3	A_2	A_3	A_4		B_2				B_3		A_{13}		A_{15}				B_1			D_1

	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
1	A_2	C_4	C_5	C_6	C_7	C_8	C_9	C_1	C_2	A_{12}	A_{13}	C_5	C_6	C_7	C_8	C_9	C_1	C_2	C_3	A_2
2	C_3	B_2	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	C_3	C_4	A_{15}	A_{16}	A_{17}	A_{18}	A_{19}	A_{20}	A_{21}	A_1	C_4
3	A_3	A_4			B_3				B_4		A_{14}		B_1				B_2			

The broadcast program perfectly satisfies our assumptions. The clients that need data item C will receive C_1, C_2, …, C_9 in at most ten time slots regardless of when the clients started listening to the broadcast channels. Moreover, in each time t, the number of empty slots is stored in *num*(t). Therefore, $0 \leq num(t) \leq m$. The server records the num list of the broadcast program to facilitate the on-demand scheduling algorithm.

When the broadcast program cannot satisfy a client's request, an explicit request can be sent to the server about the needed data item associated with the tolerable deadline via the uplink channel. The unallocated time slots are used to transmit the required data items.

3.3 On-Demand Program Generation

The main issue in developing an on-demand scheduling algorithm is scalability. That is, the scheduling overhead of processing a new request and selecting the next served request must be reduced. When new requests arrive, the server inserts them into a *priority queue*, *PQ*, according to the priorities determined by the *request insertion policies*. The on-demand scheduler selects the request with the highest priority and then broadcast the data item of interest. Assume that each request is of the form X(X.*page_set* , *X.deadline*), where *X.page_set* is the demand page set of data item *X* and *X.deadline* is the timing constraint of the request. The number of pages in *X.page_set* is denoted by *X.psize*. Moreover, the constraint that *X.psize* ≦ *X.deadline* * *m* must be met. The priority of each request is determined by the following *On-Demand Request Insertion Policies*:

Policy 1. Earliest Deadline First (EDF)

$$\text{priority } \alpha \ \frac{1}{X.deadline}.$$

Policy 2. Smallest Size First (SSF)

$$\text{priority } \alpha \ \frac{1}{X.psize}.$$

Policy 3. Largest Size First (LSF)

$$\text{priority } \alpha \ X.psize.$$

Policy 4. Smallest Difference First (SDF)

$$\text{priority } \alpha \ \frac{1}{X.deadline*m - X.psize}.$$

Policy 5. Max Frequency Times Difference First (MFTDF)

$$\text{priority } \alpha \ \frac{\text{Access Frequency of data item } X}{X.deadline*m - X.psize}.$$

Three factors considered when developing the priority of each request: the deadline, the size of the demand page set, and the access frequency. The priority in the first three request insertion policies is determined by only one factor. However, in the SDF request insertion policy, the priority is decided by the difference between *X.deadline* and *X.psize*, which can be viewed as the *tightness* of the request. Assume that the current time is *t*, and that *m* is the number of receivers. If we do not broadcast the first demand page of data item *X* during time $\left[t, \left\lfloor \frac{t*m + b*m - X.deadline}{m} \right\rfloor\right]$, we should drop this request from PQ. Therefore, the smaller the value is, the higher the priority that should be given to this request. On the other hand, many requests may require the same data item in a very short duration. Transmitting one copy of this data item will satisfy all requests. The MFTDF request insertion policy considers the access frequency of each data item together with the deadline and the size of the demand page set. The requests are inserted into PQ according to their priorities defined by each request insertion policy.

After the broadcast program is constructed, the empty time slots in each time unit are used in the on-demand mode. It is obvious that not every request can be served. Therefore, a *Request Selection Algorithm* that selects a request from PQ as the *processing request* is proposed here. The details of the algorithm are as follows:

The *Request Selection Algorithm*:

Algorithm Request Selection

For each nonzero *num*(*t*) and PQ not empty ,

Step 1: Select the first request *X*(*X.page_set*, *X.deadline*) in PQ.

```
Compute $num\_sum = num(t) + num(t + 1) + \ldots + num(t - X.deadline - 1)$.
B1: If $num\_sum < X.psize$, then remove the request from PQ. Go to step 1.
B2: If $num\_sum \geqq X.psize$, then find the minimum $t'$ such that $X.psize \leqq num(t)$
        $+ num(t + 1) + \ldots + num(t')$.
Allocate slots $= num(t) + \ldots + num(t' - 1)$.
Set $num(t) \sim num(t' - 1)$ to zero.          /* allocate $num(t) \sim num(t' - 1)$ to $X$ */
$num(t') = num(t') - (X.psize - allocated\ slots)$.    /* calculate the remaining available slots in $num(t')$ */
Remove the request from PQ and set this request as the processing request.
```

The idea behind this algorithm is that for every nonzero *num*(t), we select the first request X and check if the total number of empty slots in the period $[t, t + X.deadline - 1]$ is greater than or equal to the size of the demand page set. If the condition holds, the server broadcasts the demand pages. Otherwise, this request is removed from PQ and the next request is examined.

For example, a part of the num list of the generated broadcast program in the previous example is:

Time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
num	0	0	0	1	0	1	1	1	0	1	0	1	0	1	1	1	0	1	1	0	0

Assume that the EDF request insertion policy is used. Because there are no time slots available in $T = 0 \sim 2$, no request will be selected. In $T = 3$, the requests in PQ are $X(X_1 \rightarrow X_3, 5)$, $Y(Y_1 \rightarrow Y_6, 8)$, and $W(W_1 \rightarrow W_3, 12)$. We check that $num\_sum(X) = num(3) + num(4) + \ldots + num(7) = 4 > 3$; thus, $X(X_1 \rightarrow X_3, 5)$ is set to be the processing request. After broadcasting X_1 in $T = 3$, X_2 in $T = 5$ and X_3 in $T = 6$, we find that the num list has been modified as follows:

Time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
num	0	0	0	0	0	0	0	1	0	1	0	1	0	1	1	1	0	1	1	0	0

Afterwards, in $T = 7$, there is one empty slot available. We select another request from PQ as the processing request and proceeds.

Clients may require the same data item as the processing request. Nevertheless, if some pages of the data item have been broadcast, the clients cannot retrieve all pages of the data item. One solution to this problem is to treat every request as a new one. Obviously, this will cause a frightful waste of bandwidth. This problem can be solved as follows. Compute the differences set Δ of the demand page set of the requests in PQ which require the same data item as the processing request. If Δ is empty and the deadline of the request is larger than that of the processing request, the requests with the same required data item in PQ can be dropped. This is because these requests can be satisfied by the processing request. If Δ is not empty, the demand page set of the requests with the same required data item in PQ are set to Δ and kept in PQ even if the deadlines are smaller than that of the processing request. We cannot exclude the possibility that these requests may be satisfied. Following is our on-demand scheduling algorithm.

The *On-Demand Scheduling Algorithm*:

Algorithm On-Demand Scheduling

Setp 1: For each time *t*, remove requests with $X.psize > X.deadline * m$ in PQ.

Step 2: B1: If $num(t) = 0$, set $t = t + 1$. Subtract 1 from the deadline of the processing request and each request in PQ. Go to step1.

B2: If $num(t) > 0$, generate the processing request using the Request Selection Algorithm.

Select all requests in PQ with the same required data item as the processing request. Compute each difference set Δ of the demand page set between requests in PQ and the processing request. There are three conditions:

① Δ is ϕ, and the deadline of the request in PQ is greater than that of the processing request. Drop the request from PQ. /* can be satisfied by broadcasting processing request */

② Δ is not ϕ, and the deadline of the request in PQ is greater than or equal to that of the processing request. Keep the request in PQ and set Δ as the demand page set of the request. /* part of the demand page set "can" be satisfied by broadcasting processing request */

③ Δ is not ϕ, and the deadline of the request in PQ is smaller than the processing request. Keep the request in PQ and set Δ as the demand page set of the request. /* part of the demand page set "may" be satisfied by broadcasting processing request */

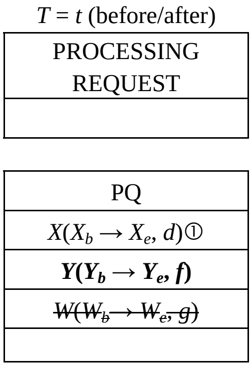

$T = t$ (before/after)
PROCESSING REQUEST
PQ
$X(X_b \rightarrow X_e, d)$①
$\mathbf{Y(Y_b \rightarrow Y_e, f)}$
~~$W(W_b \rightarrow W_e, g)$~~

Fig. 2. Graphical representation of the PQ and the processing request.

We continue with the previous example to illustrate the on-demand scheduling algorithm. Fig. 2 shows a graphical representation of PQ and the processing request. *T* (before) means the state of PQ in time *T* before allocation. The state of PQ after allocation is denoted as T (after). $X(X_b \rightarrow X_e, d)$①, X_b, and X_e indicate the beginning and end page of the demand pages of item X. d is the deadline of the request. ① means that the data item satisfies the 1st condition in B2 of step 2 in the on-demand scheduling algorithm. **$Y(Y_b \rightarrow Y_e, f)$** is the new request that arrives in time T and is shown in boldface. ~~$W(W_b \rightarrow W_e, g)$~~ means that this request is dropped for a certain reason. In this example, the request insertion policy is EDF.

In the following, a part of the resultant broadcast program and the on-demand program is shown. (The time slots with shadows are used for on-demand transmission.)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	A_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	A_{11}	A_{12}	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_1	A_1
2	C_1	B_1	D_1	A_5	A_6	A_7	A_8	A_9	A_{10}	C_1	C_2	A_{14}	B_4	A_{16}	A_{17}	A_{18}	A_{19}	A_{20}	A_{21}	C_2
3	A_2	A_3	A_4	X_1	B_2	X_2	X_3	X_1	B_3	X_2	A_{13}	…	A_{15}	…	…	…	B_1	…	…	D_1

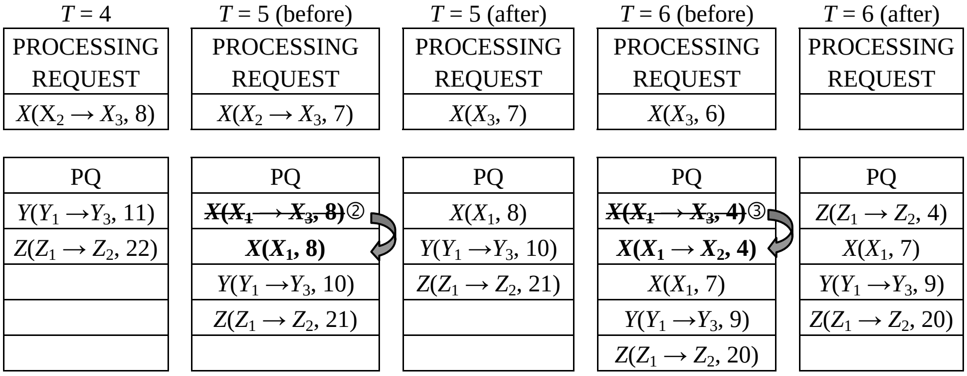
The step by step results with explanations of the on-demand scheduling algorithm are given in the following:

$T = 0$	$T = 1$	$T = 2$	$T = 3$ (before)	$T = 3$ (after)
PROCESSING REQUEST	PROCESSING REQUEST	PROCESSING REQUEST	PROCESSING REQUEST	PROCESSING REQUEST
			$X(X_1 \rightarrow X_3, 9)$	$X(X_2 \rightarrow X_3, 9)$
PQ	PQ	PQ	PQ	PQ
$X(X_1 \rightarrow X_3, 12)$	$X(X_1 \rightarrow X_3, 11)$	$X(X_1 \rightarrow X_3, 10)$	~~**$X(X_1 \rightarrow X_3, 10)$**~~ ①	$Y(Y_1 \rightarrow Y_3, 12)$
$Y(Y_1 \rightarrow Y_3, 15)$	$Y(Y_1 \rightarrow Y_3, 14)$	$Y(Y_1 \rightarrow Y_3, 13)$	$Y(Y_1 \rightarrow Y_3, 12)$	$Z(Z_1 \rightarrow Z_2, 23)$
	$Z(Z_1 \rightarrow Z_2, 25)$	$Z(Z_1 \rightarrow Z_2, 24)$	$Z(Z_1 \rightarrow Z_2, 23)$	

In $T = 0 \sim 2$, there is no empty slots; therefore, no request in PQ will be served. In $T = 3$ (before allocation), $X(X_1 \rightarrow X_3, 9)$ is selected as the processing request, and the new arrival request $X(X_1 \rightarrow X_3, 10)$ is deleted since it can be satisfied by the processing request.

In $T = 5$, the new request $X(X_1 \rightarrow X_3, 8)$ and the processing request $X(X_2 \rightarrow X_3, 7)$ are considered. The difference set Δ is not empty, and the deadline of the former request is larger than that of the later one; therefore, $X(X_1 \rightarrow X_3, 8)$ can be replaced by $X(X_1, 8)$. In $T = 6$, the difference set D of new request $X(X_1 \rightarrow X_3, 4)$ and the processing request $X(X_3, 6)$ is not empty. We replace $X(X_1 \rightarrow X_3, 4)$ with $X(X_1 \rightarrow X_2, 4)$. Although the deadline of the later request is smaller than that of the processing request, this request can be satisfied if it is selected as the processing request in $T = 7$.

As we follow the above procedure, the on-demand program is generated. Moreover, the steps in the on-demand scheduling algorithm with other request insertion policies are similar to those of EDF, and are omitted here.



$T = 4$	$T = 5$ (before)	$T = 5$ (after)	$T = 6$ (before)	$T = 6$ (after)
PROCESSING REQUEST	PROCESSING REQUEST	PROCESSING REQUEST	PROCESSING REQUEST	PROCESSING REQUEST
$X(X_2 \rightarrow X_3, 8)$	$X(X_2 \rightarrow X_3, 7)$	$X(X_3, 7)$	$X(X_3, 6)$	

PQ	PQ	PQ	PQ	PQ
$Y(Y_1 \rightarrow Y_3, 11)$	~~$\boldsymbol{X(X_1 \rightarrow X_3, 8)}$~~②	$X(X_1, 8)$	~~$\boldsymbol{X(X_1 \rightarrow X_3, 4)}$~~③	$Z(Z_1 \rightarrow Z_2, 4)$
$Z(Z_1 \rightarrow Z_2, 22)$	$\boldsymbol{X(X_1, 8)}$	$Y(Y_1 \rightarrow Y_3, 10)$	$\boldsymbol{X(X_1 \rightarrow X_2, 4)}$	$X(X_1, 7)$
	$Y(Y_1 \rightarrow Y_3, 10)$	$Z(Z_1 \rightarrow Z_2, 21)$	$X(X_1, 7)$	$Y(Y_1 \rightarrow Y_3, 9)$
	$Z(Z_1 \rightarrow Z_2, 21)$		$Y(Y_1 \rightarrow Y_3, 9)$	$Z(Z_1 \rightarrow Z_2, 20)$
			$Z(Z_1 \rightarrow Z_2, 20)$	

4. SIMULATION

4.1 Parameters Setting

The following table shows the simulation parameters (Table 1):

Table 1. Simulation parameters.

Parameter	Default Value	Range
Request arrival rate	Poisson distribution among 0 ~ 50 requests	0 ~ 80 request/time unit
Size of each item	Uniform distribution among 1 ~ 8 pages	1 ~ 8 pages
Number of channels	8 channels	-
Number of receivers	4	-
Number of data items	500	200 ~ 1000
RF value	0	0 ~ 1

We considered a database consisting of 500 distinct data items with various sizes and periods. The period and size of each data item followed the uniform distribution and were permanently associated with each data item. The access frequency distribution of the data items was modeled as a Zipf distribution. Moreover, a workload generator generated on-demand requests associated with deadline constraints in every time unit. The request arrival rate followed a Poisson distribution.

Unlike the traditional broadcast scheduling algorithms that try to minimize the average access time, the metric used to evaluate the performance of our scheduling algorithms is the percentage of deadlines of requests that are missed.

4.2 Comparison of the Scheduling Time with Different Broadcast Cycle Lengths

After selecting a feasible, optimal broadcast data set is selected, the broadcast program is generated offline. The broadcast cycle length is determined by the least common multiple of all the periods of data items in the broadcast data set. Fig. 3 shows the

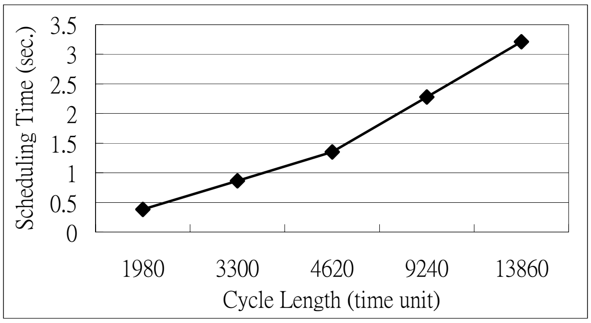


Fig. 3. The scheduling time versus different broadcast cycle lengths.

relationship between the time needed to generate the periodic broadcast program and different broadcast cycle lengths.

It can be seen that as the length increases, the time needed to compute the broadcast program increases.

4.3 Comparison of Different Request Insertion Policies

To evaluate the performance of five request insertion policies, two sets of experiments were performed.

In the first simulation, we examined the effects of different access frequency distributions. The access frequency of each data item followed a Zipf distribution, which can be expressed as $P_i = \frac{(1/i)^\theta}{\sum_{i=1}^{M}(1/i)^\theta}, 1 \le i \le M,$ where M is the number of data items and θ is the *access skew coefficient*. For $\theta = 0$, the Zipf distribution reduces to a uniform distribution. As θ increases, the distribution becomes more skewed, which means that the range of p_i becomes larger. By tuning the access skew coefficient, we can get different access frequency models. Figs. 4 and 5 plot the percentage of missed deadlines over all requests and over on-demand requests versus different access skew coefficients. From the simulation results, we observe that as θ increases, more requests are served. This is because when requests target to fewer data items, transmission of these hot data items will satisfy more requests. Moreover, for $\theta = 0$, more requests will be sent to the server than is the case with the skewed model since data items are accessed with equal probability.

In the second simulation, we examined the effects of different workloads. The workload of the server is determined by the number of requests that arrive per time unit. The time needed to insert a new request into PQ is shown in Fig. 6. The time spent implementing the EDF insertion policy is shorter than that for all the other request insertion policies. The reason is that as the size of the demand page set and the access frequency of requests changes, the order of all the requests in PQ will be affected in the other four request insertion policies. Since the time needed to run the insertion algorithm is relatively small, we can ignore it.

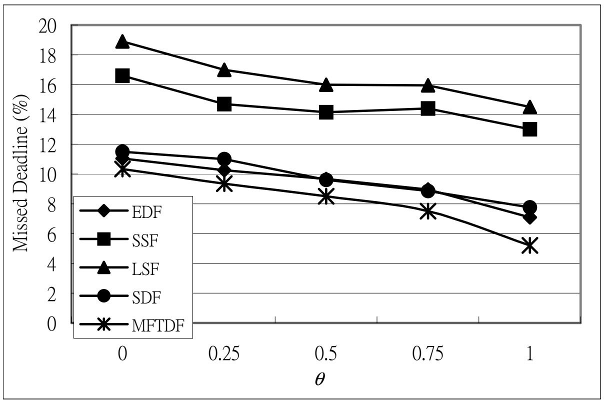


Fig. 4. The missed deadline over all requests versus different access skew coefficient.

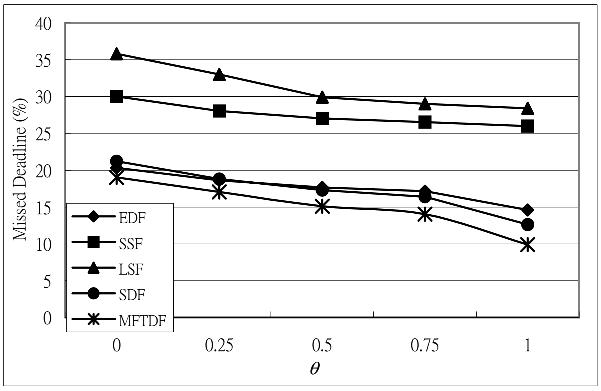


Fig. 5. The missed deadline over on-demand requests versus different access skew coefficient.

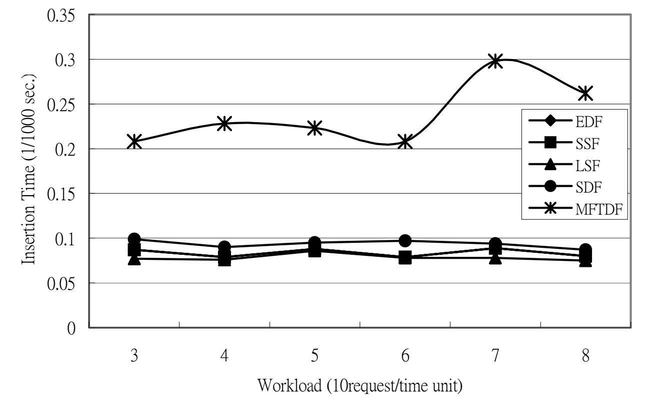


Fig. 6. The request insertion time versus different workload.

4.4 Comparison of Different Access Pattern Models

Most of the previous researches on data broadcasting assumed that the access patterns of mobile clients are known and do not change over time. However, in real applications, clients may move from cell to cell. In order to model the stability of access patterns, a parameter named *random factor* (RF) is introduced. RF is a value between 0 and 1. The larger the RF value is, the lower is the probability of the clients following the given access pattern model.

In the following simulation, the effect of changing access patterns was evaluated. The simulation procedure was as follows: First, a set of requests was derived as the history access pattern model. Based on this access pattern model, the broadcast program was generated. Then, a set of requests was generated according to the new access pattern model with a given RF value. In particular, RF = 0 means the clients never violates the given access pattern model, while RF = 1 means the clients never follows the given access pattern model. We recorded the percentage of missed deadlines versus various RF values and depict the results in the following figures.

Figs. 7 and 8 show the percentages of missed deadlines over all requests and over on-demand requests, respectively. It can be observed that the changing access patterns have a great influence on the performance of the system. For smaller RF values, more requests can be answered by the periodic broadcast program; consequently, fewer requests will be sent to the server than is the case with access models with larger RF values. As more requests are sent to the server, the workload of the server increases, which results in more requests missing their deadlines. Therefore, by collecting the real access patterns of the clients after a certain number of broadcast cycles, the broadcast scheduler can generate more effective broadcast programs.

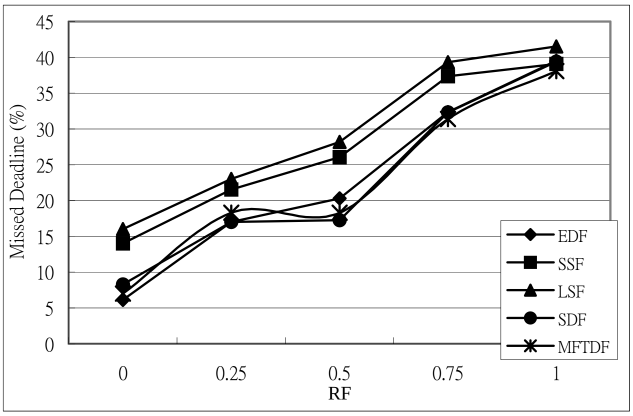


Fig. 7. The missed deadline over all requests versus different RF values.

5. CONCLUSIONS AND FUTURE WORKS

In this paper, the problem of real-time data scheduling in multiple broadcast channel environments where clients are equipped with multiple receivers has been considered.

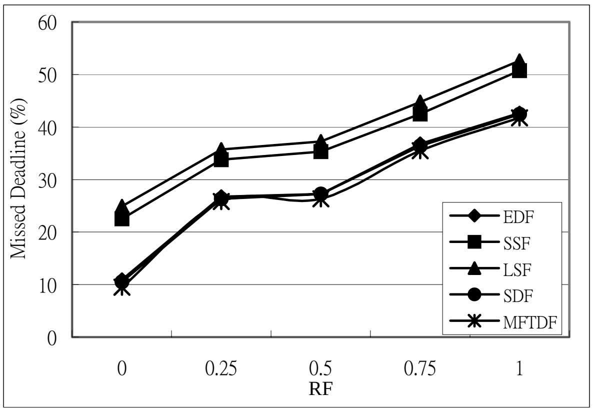


Fig. 8. The missed deadlines over on-demand requests versus different RF values.

The data items in the database server are classified into a broadcast data set and an on-demand data set. By transforming the broadcast program generation problem from the MCMR environment into that in the SCSR environment, we can construct a periodic broadcast program satisfying the timing constraint. Moreover, an on-demand program generation algorithm has been proposed.

Simulations were performed to verify our approach. From the simulation results, we find that MFTDF, SDF or EDF achieve good performance. Moreover, the time needed to insert requests into the PQ of the proposed policy is also short.

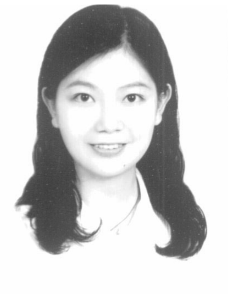
REFERENCES

1. S. M. Alamouti, "A simple transmit diversity technique for wireless communications," *IEEE Journal on Selected Areas in Communications*, Vol. 16, 1998, pp. 1451-1458.
2. S. Acharya, R. Alonso, M. Franklin, and S. Zdonik, "Broadcast disks: data management for asymmetric communication environments," in *Proceedings of ACM-SIGMOD International Conference on Management of Data*, 1995, pp. 199-210.
3. S. Acharya, R. Alonso, and S. Zdonik, "Dissemination-based data delivery using broadcast disks," *IEEE Personal Communications*, Vol. 2, 1995, pp. 50-56.
4. D. Aksoy and M. Franklin, "Scheduling for large-scale on-demand data broadcasting," in *Proceedings of the 1998 IEEE INFOCOM Conference*, 1998, pp. 651-659.
5. S. Acharya, M. Franklin, and S. Zdonik, "Balancing push and pull for data broadcast," in *Proceedings of ACM-SIGMOD International Conference on Management of Data*, 1997, pp. 183-194.
6. S. Baruah and A. Bestavros, "Pinwheel scheduling for fault tolerant broadcast disks in real-time database systems," in *Proceedings of 13th International Conference on Data Engineering*, 1997, pp. 543-551.
7. S. Baruah, N. Cohen, G. Plaxton, and D. Varvel, "Proportionate progress: a notion of fairness in resource allocation," *Algorithmica*, Vol. 15, 1996, pp. 600-625.

8. S. Baruah and S. Lin, "Improved scheduling of generalized pinwheel task systems," in *Proceedings of 4th International Workshop on Real-Time Computer Systems Applications*, 1997, pp. 73-79.
9. S. Baruah and S. S. Lin, "Pfair scheduling of generalized pinwheel task systems," *IEEE Transactions on Computers*, Vol. 4, 1998, pp. 812-816.
10. C. H. Hsu, G. Lee, and A. L. P. Chen, "A near optimal algorithm for generating broadcast programs on multiple channels," *the 10th International Conference on Information and Knowledge Management* (*CIKM 2001*), 2001, pp. 303-309.
11. J. Fernandex and K. Ramaritham, "Adaptive dissemination of data in time-critical asymmetric communication environments," *EuroMicro Conference on Real-Time Systems*, 1997, pp. 195-203.
12. C. H. Hsu, G. Lee, and A. L. P. Chen, "Index and data allocation on multiple broadcast channels considering data access frequencies," in *Proceedings of 3rd International Conference on Mobile Data Management* (*MDM 2002*), 2002, pp. 87-93.
13. R. W. Heath and A. J. Paulraj, "Linear dispersion codes for MIMO systems based on frame theory," *IEEE Transactions on Signal Processing*, Vol. 50, 2002, pp. 2429-2441.
14. R. Holte, A. Mok, L. Rosier, I. Tulchinsky, and D. Varvel, "The pinwheel: a real-time scheduling problem," in *Proceedings of 22nd International Conference on System Science*, 1989, pp. 693-702.
15. S. Hameed and N. H. Vaidya, "Efficient algorithms for scheduling data broadcast," *ACM-Baltzer Wireless Networks*, Vol. 5, 1999, pp. 183-193.
16. T. Imielinski, S. Viswanathan, and B. R. Badrinath, "Data on air: organization and access," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 9, 1997, pp. 353-372.
17. S. Jiang and N. Vaidya, "Scheduling data broadcast to impatient users," in *Proceedings of International Workshop on Data Engineering for Wireless and Mobile Access*, 1999, pp. 52-59.
18. S. C. Lo and A. L. P. Chen, "Optimal index and data allocation in multiple broadcast channels," in *Proceedings of IEEE International Conference on Data Engineering*, 2000, pp. 293-302.
19. G. J. Li and B. W. Wah, "Systolic processing for dynamic programming problems," in *Proceedings of the International Conference on Parallel Processing*, 1985, pp. 434-441.
20. G. Lee, Y. N. Pan, and A. L. P. Chen, "Scheduling real-time data items in multiple channels and multiple receivers environments," in *Proceedings of IEEE International Conference on Distributed Computing Systems*, 2002, pp. 455-456.
21. N. Shivakumar and S. Venkatasubramanian, "Energy-efficient indexing for information dissemination in wireless systems," *ACM Journal of Wireless and Nomadic Application*, Vol. 1, 1996, pp. 433-446.
22. K. L. Tan and J. Cai, "Batch Scheduling for demand-driven servers in wireless environments," *Journal of Information Sciences*, Vol. 109, 1998, pp. 281-298.
23. V. Tarokh, N. Seshadri, and A. R. Calderbank, "Space-time codes for high data rate wireless communication: performance criterion and code construction," *IEEE Transactions on Information Theory*, Vol. 44, 1998, pp. 744-765.
24. P. Xuan, S. Sen, O. Gronzalez, J. Fernandex, and K. Ramaritham, "Broadcast on

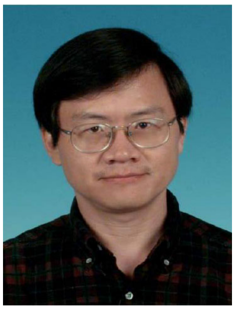
demand: efficient and timely dissemination of data in mobile environments," in *Proceedings of 3rd IEEE Symposium on Real-Time Technology and Application Symposium*, 1997, pp. 38-48.

25. A. C. C. Yao, "On the parallel computation for the knapsack problem," in *Proceedings of 13th Annual ACM Symposium on Theory of Computing*, 1981, pp. 123-127.
26. L. Zheng and D. N. C. Tse, "Diversity and multiplexing: a fundamental tradeoff in multiple antenna channels," *IEEE Transactions on Information Theory*, Vol. 48, 2002, pp. 359-383.



Guanling Lee (李官陵) received the B.S., M.S, and Ph.D. degrees in Computer Science from National Tsing Hua University, Taiwan, R.O.C., in 1995, 1997, and 2001, respectively. She joined National Dong Hwa University, Taiwan, as an Assistant Professor in the Department of Computer Science and Information Engineering in August 2001. Her research interests include resource management in the mobile environment, data scheduling on wireless channels, search in the P2P network, and data mining.

Yi-Ning Pan (潘依寧) received the M.S. degrees in Computer Science from National Tsing Hua University, Taiwan, R.O.C., in 2001. She joined Springsoft Inc., as an Engineer in August 2001.



Arbee L. P. Chen (陳良弼) received the B.S. degree in Computer Science from National Chiao Tung University, Taiwan in 1977, and the Ph.D. degree in Computer Engineering from the University of Southern California in 1984. He was a Member of Technical Staff at Bell Communications Research, New Jersey, from 1987 to1990, an Adjunct Associate Professor in the Department of Electrical Engineering and Computer Science, Polytechnic University, New York, and a Research Scientist at Unisys, California, from 1985 to 1986. He joined National Tsing Hua University (NTHU), Taiwan, as a National Science Council (NSC) sponsored Visiting Specialist in August 1990, and became a Professor of the Department of Computer Science, NTHU, in 1991. In August 2001 he took a leave from NTHU and assumed the position of the Chairman of the Department of Computer Science and Information Engineering at National Dong Hwa University, Taiwan. Since August 2004, he became Chengchi University Chair Professor at National Chengchi University, Taiwan. His current research interests include multimedia databases, data mining and data stream management systems.

Dr. Chen organized (and served as a Program Co-Chair for) 1995 IEEE Data Engineering Conference, 1999 International Conference on Database Systems for Advanced

Applications (DASFAA) and 2002 International Computer Symposium, and served as a Conference Co-Chair for 2002 Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), and Program Co-Chair for 2000 PAKDD and 1997 International Conference on Cooperative Information Systems (CooPIS). He was invited to deliver a speech on Music Representation, Indexing and Retrieval at the NSF sponsored Inaugural International Symposium on Music Information Retrieval (ISMIR) at Plymouth, USA, 2000, and delivered a tutorial on Content-based Music Retrieval and Analysis at 2003 ISMIR. He also delivered a tutorial on Multimedia Database Systems at 1999 DASFAA. He was a visiting scholar at the Center for Computer Assisted Research in the Humanities/Center for Computer Research in Music and Acoustics, Stanford University, January 2003, and at Department of Computer Science, Stanford University, January 2004. He is an editor of several international journals including World Wide Web: Internet and Web Information Systems, Kluwer Academic Publishers. He has been a recipient of the NSC Distinguished Research Award since 1996, and is currently an NSC Fellow.