

# Dynamic Generation of Data Broadcasting Programs for a Broadcast Disk Array in a Mobile Computing Environment

Wen-Chih Peng and Ming-Syan Chen  
Department of Electrical Engineering  
National Taiwan University  
Taipei, Taiwan, ROC

E-mail: {mschen@cc.ee.ntu.edu.tw, wcpeng@arbor.ee.ntu.edu.tw}

## ABSTRACT

We explore in this paper the problem of generating hierarchical broadcast programs with the data access frequencies and the number of broadcast disks in a broadcast disk array given. Specifically, we first transform the problem of generating hierarchical broadcast programs into the one of constructing a channel allocation tree with variant-fanout. By exploiting the feature of tree generation with variant-fanout, we develop a heuristic algorithm  $VF^K$  to minimize the expected delay of data items in the broadcast program. Performance of these algorithms is analyzed. It is shown by our simulation results that by exploiting the feature of variant-fanout in constructing the channel allocation tree, the solution obtained by algorithm  $VF^K$  is of very high quality and is in fact very close to the optimal one.

*Keywords:* Broadcast disks, mobile computing, broadcast programs, multiple broadcast channels.

## 1. INTRODUCTION

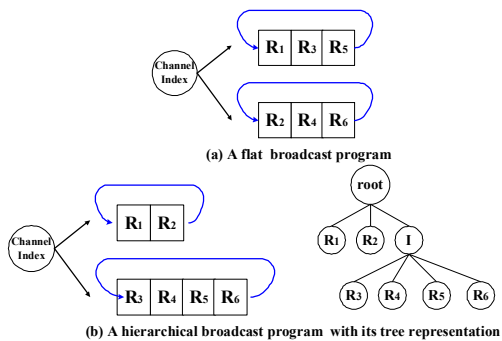
Several applications in a mobile computing environment, such as stock activities, traffic reports and weather forecast, have become increasingly popular in recent years [17][18]. It is noted that mobile computers use small batteries for their operations without directly connecting to any power source, and the bandwidth of wireless communication is in general limited. As a result, an important design issue in a mobile system is to conserve the energy and communication bandwidth of a mobile unit while allowing mobile users of the ability to access information from anywhere at anytime [3][5][10].

In order to conserve the energy and communication bandwidth of a mobile computing system, a data delivery architecture in which a server continuously and repeatedly broadcasts data to a client community through a broadcast channel was proposed in [1]. In a push-based information system, a server generates a broadcast program to broadcast data to the mobile users. This broadcast channel is

referred to as a “broadcast disk” from which mobile clients can retrieve data. The mobile users need to wait for the data of interest to appear on the broadcast channel, and the corresponding waiting time is called the expected delay of that data item. One objective of designing proper data allocation in the broadcast disks is to reduce the average expected delay of data items. Broadcasting schemes in this context have been previously addressed by other researchers [2][8][9][11][13][15][16]. Also, a significant amount of research effort has been elaborated on developing the index mechanism in multiple broadcast channels [12][14]. A system of multiple broadcast channels can be viewed as a *broadcast disk array*. The broadcast disks in a broadcast disk array can be categorized according to the *speed* of broadcast disks, where the speed of a broadcast disk corresponds to the expected delay for the data items in that broadcast disk. The data items in each broadcast disk are sent out in a round robin manner. Clearly, as the number of data items in a broadcast disk increases, the expected delay of those data items increases. As a result, the data items that are more frequently requested by mobile users should be put in fast broadcast disks, whereas cold data items can be pushed to slow broadcast disks to minimize the average expected delay of data items in the broadcast disk array. Organizing data in a broadcast disk array raises a number of new research problems. The most important issue is to develop algorithms to allocate data items to the broadcast disk array according to their access frequencies so as to minimize the average expected delay of data items. This is the very problem that we shall address in this paper.

Consider the illustrative example in Figure 1 where two broadcast programs are presented. Assume that the data items  $R_i$ ,  $1 \leq i \leq 6$ , are of the same size and Figure 1a shows a flat broadcast program, where data items are evenly allocated to the two broadcast disks and the speeds of two broadcast disks in the broadcast disk array are the same, meaning that the expected delays for all data items are equal from one to another. In contrast, Figure 1b shows another broadcast program with its channel allocation tree (or abbreviated as allocation tree). As will be described in Section 2 later, the depth of the allocation trees corresponds to the number of broadcast disks, and those leaf nodes in the same level of the allocation tree correspond to those data items to be put in the same broadcast disk. In Figure 1b, the upper channel is allocated with two data items and the lower channel is allocated with four data items. Notice that in Figure 1b the data items in the fast disk (i.e., the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.



**Figure 1: Two broadcast programs for the broadcast disk array of two broadcast disks.**

upper broadcast channel) spin twice as fast as those data items in the slow disk (i.e., the lower broadcast channel). The allocation tree in Figure 1b is called an allocation tree with variant-fanout in this paper. Denote the expected delay and the access frequency of data item  $R_i$ , respectively, as  $d_{R_i}$  and  $P_r(R_i)$ . Table 1a shows the expected delay for data items under the two different allocations of broadcast arrays, and four sets of access frequencies of data items are given in Table 1b for illustrative purposes. The average expected delay in Table 1b is obtained by multiplying the access frequency of each data item by the expected delay of that data item and summing up the results, i.e.,  $\sum_{i=1}^6 d_{R_i} * P_r(R_i)$ . Same as in [1], the expected delay for each data item in the broadcast disk  $i$  is formulated as  $\sum_{x=1}^{N_i} \frac{(N_i-x)}{N_i}$ , where  $N_i$  is the number of data items allocated in the broadcast disk  $i$ . For example, for the allocation in Figure 1a,  $N_1 = 3$  and  $N_2 = 3$ , and for the allocation in Figure 1b,  $N_1 = 2$  and  $N_2 = 4$ . The details of formulating the average expected delay can be found in [1]. It can be verified that the expected delays of data items  $R_1$  and  $R_2$  in Figure 1a are both equal to  $d_{R_1} = d_{R_2} = \frac{2+1+0}{3} = 1$ . On the other hand, the expected delays of data items  $R_1$  and  $R_3$  in Figure 1b are  $d_{R_1} = \frac{1+0}{2} = 0.5$  and  $d_{R_3} = \frac{3+2+1+0}{4} = 1.5$ , respectively.

From Table 1b and Table 1c, it is noted that when the access frequencies of all data items are the same, the flat broadcast program in Figure 1a is the one to use, and we do not want to allocate different numbers of data items to different broadcast disks as in Figure 1b. Explicitly, the average expected delay of data items for Case 1 is  $\sum_{i=1}^6 d_{R_i} * P_r(R_i) = 1*0.167+1*0.167+1*0.167+1*0.167+1*0.167+1*0.167=1$ , whereas that in Figure 1b is  $\sum_{i=1}^6 d_{R_i} * P_r(R_i) = 0.5*0.167+0.5*0.167+1.5*0.167+1.5*0.167+1.5*0.167+1.5*0.167=1.16$ . It can be verified that the average expected delay resulting from the flat broadcast program will remain the same as the access frequencies of data items vary. However, when the access frequencies become increasingly skewed, the average expected delay of allocation with variant-fanout is reduced from 1.16 in Case 1 to 0.7 in Case 4. With the access frequencies being skewed, the average expected delay of the hierarchical broadcast program based on variant-fanout becomes much smaller than that of the flat broadcast program, showing the very advantage of exploiting the feature of hierarchy in designing broadcast programs for a broadcast disk array.

Consequently, we explore in this paper the issue of gener-

Expected Delay of Data Items						
	$d_{R_1}$	$d_{R_2}$	$d_{R_3}$	$d_{R_4}$	$d_{R_5}$	$d_{R_6}$
in Figure 1a	1	1	1	1	1	1
in Figure 1b	0.5	0.5	1.5	1.5	1.5	1.5

(a)

	Access Frequency					
	$P_r(R_1)$	$P_r(R_2)$	$P_r(R_3)$	$P_r(R_4)$	$P_r(R_5)$	$P_r(R_6)$
Case 1	0.167	0.167	0.167	0.167	0.167	0.167
Case 2	0.25	0.25	0.125	0.125	0.125	0.125
Case 3	0.3	0.3	0.1	0.1	0.1	0.1
Case 4	0.4	0.4	0.05	0.05	0.05	0.05

(b)

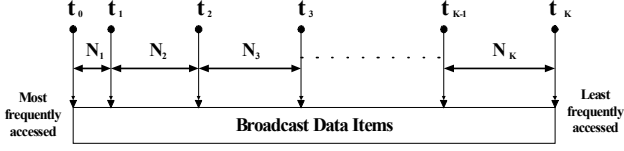
	Average Expected Delay	
	in Figure 1a	in Figure 1b
Case 1	1	1.16
Case 2	1	1
Case 3	1	0.9
Case 4	1	0.7

(c)

Table 1. Expected delays and access frequencies of data items under two broadcast programs.

ating hierarchical broadcast programs with the data access frequencies and the number of broadcast disks in a broadcast disk array given. Specifically, we first formulate the problem of generating hierarchical broadcast programs and transform this problem into the one of constructing a channel allocation tree with variant-fanout. A hierarchical broadcast program for the broadcast disk array of  $K$  broadcast disks can then be represented by a channel allocation tree with a height of  $K$ . By exploiting the feature of tree generation with variant-fanout, we develop a heuristic algorithm  $VF^K$  (standing for Variant-Fanout with the constraint  $K$ ), which is basically a family of algorithms with different values of  $K$ , to minimize the expected delay of the corresponding broadcast program. Performance of these algorithms is analyzed and sensitivity analysis on several parameters, including the number of data items and the number of broadcast disks, is conducted. It is shown by our simulation results that by exploiting the feature of variant-fanout in constructing the channel allocation tree, the solution obtained by algorithm  $VF^K$  is of very high quality. It is worth mentioning that even when the number of broadcast channels  $K$  changes dynamically, algorithm  $VF^K$  can reach the new configuration very efficiently with the number of data items required to be moved around broadcast channels minimized, showing another advantage of algorithm  $VF^K$ . It is also shown by our experimental results that algorithm  $VF^K$  is not only able to produce the solutions of very high quality but also of good scalability which is important for  $VF^K$  to be of practical use to generate hierarchical broadcast programs dynamically in a mobile computing environment.

A significant amount of research effort has been elaborated upon issues of data broadcast [1][9][11]. We mention in passing that the authors in [1] explored a push-based data delivery architecture using the broadcast disk to meet the need of mobile applications. The authors in [7] proposed



**Figure 2: Generating hierarchical broadcast programs for a broadcast disk array of  $K$  broadcast disks, where  $N_i$  is the number of data items allocated to disk  $i$  and  $t_i = \sum_{j=1}^i N_j$ .**

a mechanism to capture data access patterns which can be utilized in a data delivery model composed of push-based and pull-based delivery methods. Without fully exploiting the advantage of multiple broadcast channels, the attention of those studies in [1][4][7][12] was mainly paid to the design of data delivery model and index methods for a single broadcast channel, but not for multiple broadcast channels. The design of index methods in multiple broadcast channels was addressed in [12][14], where, however, the problem of generating hierarchical broadcast programs was not considered.

This rest of this paper is organized as follows. Preliminaries are given in Section 2. In Section 3, we develop algorithm  $VF^K$  for allocating data items to a broadcast disk array. Performance studies are conducted in Section 4. This paper concludes with Section 5.

## 2. PRELIMINARIES

Note that as the number of data items in a broadcast disk increases, the expected delay of those data items within the broadcast disk increases. Theoretically, generating such a broadcast program can be viewed as a partition problem for data items. Given the number of broadcast disks in a disk array and the access frequencies of all data items, we shall determine the proper set of data items that should be allocated to each broadcast disk in a broadcast disk array with the purpose of minimizing the average expected delay of all data items. Table 2 shows the description of symbols used in modelling the program. Figure 2 shows the program formulation of generating a hierarchical broadcast program for a broadcast disk array of  $K$  broadcast disks. Denote the total number of data items as  $n$ , and a data item as  $R_i$ ,  $1 \leq i \leq n$ . Recall that  $P_r(R_i)$  is the access frequency of  $R_i$  and  $\sum_{i=1}^n P_r(R_i) = 1$ . Assume that all data items in Figure 2 have been sorted according to the descending order of the access frequencies of all data items. As can be seen in Figure 2, a broadcast disk array of  $K$  broadcast disks corresponds to the partition of  $n$  data items into  $K$  groups, where  $N_i$  is the number of data items to be allocated to broadcast disk  $i$ . We then have  $\sum_{i=1}^K N_i = n$ . According to the characteristics of broadcast disks, we have the following properties.

**Property 1:** The expected delay of the data items within the broadcast disk  $i$ , denoted by  $d_i$ , is  $d_i = \sum_{j=1}^{N_i} \frac{N_i - j}{N_i}$ , where  $N_i$  is the number of data in that broadcast disk.

**Property 2:** Let  $t_i = \sum_{h=1}^i N_h$  and  $t_0 = 0$ . Then,  $N_i$  data items, denoted by  $R_j$ ,  $t_{i-1} + 1 \leq j \leq t_i$ , are allocated to broadcast disk  $i$ , and  $d_i = d_{R_j}$  for  $j \in [t_{i-1} + 1, t_i]$ .

For example, in Figure 1b,  $d_1 = d_{R_1} = d_{R_2} = 0.5$  since  $R_1$  and  $R_2$  are in broadcast disk 1, and  $d_2 = d_{R_3} = d_{R_4} =$

Description	Symbol
Number of broadcast disks in a broadcast disk array	$K$
Number of data items within broadcast disk $i$	$N_i$
The expected delay of data items within broadcast disk $i$	$d_i$
The $j$ th data item	$R_j$
The access frequency of data item $R_j$	$P_r(R_j)$
The expected delay of data item $R_j$	$d_{R_j}$
The number of leaf nodes in level $i$ of the allocation tree	$L_i$
The weight of leaf nodes in level $i$ of the allocation tree	$w_i$
The aggregate access frequency for the leaf nodes in level $i$	$P_{L_i}$

Table 2. Description of symbols

$d_{R_5} = d_{R_6}$  since  $R_3, R_4, R_5$  and  $R_6$  are in broadcast disk 2. The problem that we study in this paper can be formally defined as follows:

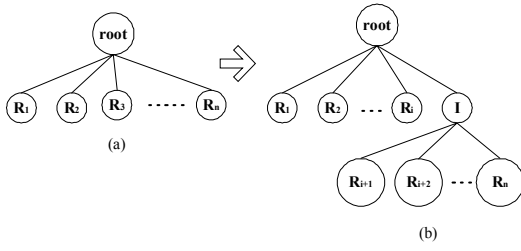
**Problem of generating a hierarchical broadcast program:** Given the number of broadcast disks in a disk array and the access frequencies of all data items, we shall determine the proper set of data items that should be allocated to each broadcast disk in a broadcast disk array with the purpose of minimizing the average expected delay of all data items. The average expected delay of all data items in a broadcast disk array of  $K$  broadcast disks can be formulated as follows:

$$\begin{aligned} \sum_{j=1}^n d_{R_j} * P_r(R_j) &= \sum_{i=1}^K d_i \sum_{j=t_{i-1}+1}^{t_i} P_r(R_j) \\ &= \sum_{i=1}^K \left( \sum_{q=1}^{N_i} \frac{N_i - q}{N_i} \right) \sum_{j=t_{i-1}+1}^{t_i} P_r(R_j), \end{aligned}$$

where  $t_i = \sum_{h=1}^i N_h$  and  $t_0 = 0$ .

**Problem transformation:** As mentioned above, generating such a broadcast program can be viewed as a partition problem. We now would like to transform this partition problem on data items into the one of generating a channel allocation tree. The initial allocation tree is mainly a tree of all data items attached to the root node (as shown in Figure 3a). Note that the leaf nodes in the same level of the allocation tree correspond to a set of data items to be put in the same broadcast disk. Thus, the allocation tree in Figure 3a corresponds to the case that all data items are put in one broadcast disk. We next expand the allocation tree by moving a set of nodes to one level lower as shown in Figure 3b (the criterion to determine such a set of nodes will be explained in Section 3 later). Figure 3b corresponds to the case that two broadcast disks are used to store the data items. This procedure continues by performing more partitions on data items until all broadcast disks in the broadcast disk array are used. As such, a hierarchical broadcast program for a broadcast disk array of  $K$  broadcast disks can be represented as a channel allocation tree with a height of  $K$ .

It can be seen that similar to the formulation of the ex-



**Figure 3: Grouping a set of nodes and moving them to a lower level to reduce the weights of leaf nodes.**

pected delay for those data items residing in a broadcast disk, the weight of leaf nodes in level  $i$ , denoted by  $w_i$ , can be formulated as  $w_i = \sum_{x=1}^{L_i} \frac{(L_i - x)}{L_i}$ , where  $L_i$  is the number of leaf nodes in the level  $i$ . The weight  $w_i$  in fact corresponds to the expected delay of each data item in level  $i$ . Explicitly, when the construction of an allocation tree is completed,  $L_i$  equals  $N_i$  and  $w_i$  equals  $d_i$ . Let  $P_{L_i}$  be the summation of access frequencies of data items associated with leaf nodes in level  $i$ . For example, it can be seen from Figure 1b that the depth of the allocation tree is 2 and those leaf nodes in the same level are allocated in the same broadcast disk. For that allocation tree with access frequencies given in Case 3 of Table 1b, we have  $w_1 = 0.5$ ,  $w_2 = 1.5$ ,  $P_{L_1} = 0.6$ , and  $P_{L_2} = 0.4$ . According to the formulation of average expected delay for broadcast disk array, we have the cost of the tree below:

$$\sum_{i=1}^K w_i P_{L_i} = \sum_{i=1}^K \left( \sum_{q=1}^{L_i} \frac{L_i - q}{L_i} \right) * P_{L_i},$$

$$\text{where } P_{L_i} = \sum_{j=s_{i-1}+1}^{s_i} P_r(R_j), \quad s_i = \sum_{h=1}^i L_h \text{ and } s_0 = 0.$$

As such, the problem of generating a hierarchical broadcast program can be transformed into the one of building an allocation tree with the minimal cost.

### 3. ALGORITHM VF<sup>K</sup>

We devise in Section 3.1 a heuristic algorithm VF<sup>K</sup> to generate the channel allocation tree which explores the feature of tree generation with variant-fanout to minimize the expected delay of the corresponding broadcast program. The execution scenario of algorithm VF<sup>K</sup> is illustrated in Section 3.2.

#### 3.1 Design of Algorithm VF<sup>K</sup>

With the problem transformation described above, we devise algorithm VF<sup>K</sup> (standing for Variant-Fanout with the constraint K) to explore variant-fanout in the allocation tree generation to minimize the cost of this tree.

Algorithm VF<sup>K</sup> is greedy in nature and builds the allocation tree in a top down manner. Algorithm VF<sup>K</sup> starts with attaching all data records to the root node. Then, after some evaluation, algorithm VF<sup>K</sup> groups nodes with small access frequencies and moves them to one level lower so as to re-

duce the cost of the tree. Figure 3 shows the scenario of grouping a set of nodes and moving them to a lower level.

**Definition 1:** Suppose that level  $v$  in the allocation tree has  $j - i + 1$  data nodes,  $R_i, R_{i+1}, \dots, R_j$ . The cost of level  $v$  is defined as  $C_{i,j} = \sum_{k=1}^{j-i+1} \frac{(j-i+1)-k}{j-i+1} \sum_{q=i}^j P_r(R_q)$ , which is equal to  $w_v * P_{L_v}$ , where  $w_v$  and  $P_{L_v}$  are, respectively, the weight of leaf nodes and the aggregate access frequency for the leaf nodes in level  $v$  of the allocation tree.

In essence, the value of  $C_{i,j}$  is related to the average expected delay of leaf nodes in level  $v$ . It can be seen from Figure 3 that moving down those nodes to the next level decreases the corresponding expected delay of leaf nodes, and thus the cost of the allocation tree will be reduced. In order to evaluate the reduction of partitioning, we have the following definition.

**Definition 2:** Suppose that node  $R$  has  $j - i + 1$  child data nodes,  $R_i, R_{i+1}, \dots, R_j$ , which are sorted according to the descending order of  $P_r(R_q)$ ,  $i \leq q \leq j$ , i.e.,  $P_r(R_q) \geq P_r(R_y)$  iff  $q \leq y$ . The reduction gain achieved by grouping nodes  $R_{p+1}, R_{p+2}, \dots, R_j$  and attaching them under a new child node, denoted by  $\delta(p)$ , can be formulated as  $\delta(p) = C_{i,j} - (C_{i,p} + C_{p+1,j})$ .

In light of Definition 2, we devise algorithm VF<sup>K</sup> below which contains a procedure *Partition* to identify the group of nodes to be moved downward in each execution level so as to maximize the reduction gain in each step.

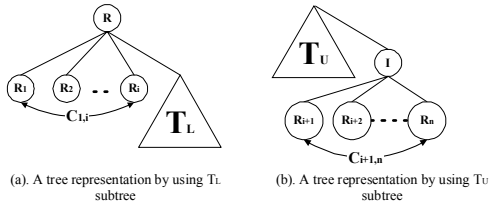
Algorithm VF<sup>K</sup>:

**Input:** Assume that  $R_1, R_2, \dots, R_n$  have been sorted according to the descending order of  $P_r(R_j)$ ,  $1 \leq j \leq n$ , i.e.,  $P_r(R_q) \geq P_r(R_y)$  iff  $q \leq y$ .  $K$  is the number of broadcast disks in a broadcast disk array.

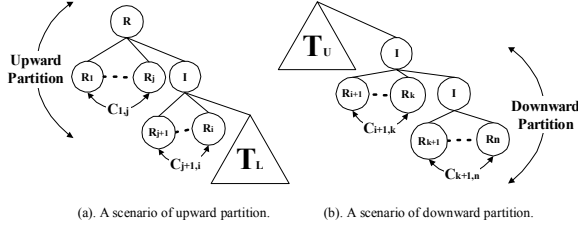
**Output:** The resulting allocation tree.

**begin**

1. Create Table AT with K rows;
2.  $AT(1).B = 1$ ;
3.  $AT(1).E = n$ ;
4.  $AT(1).LC = C_{1,n}$ ;
5. for each row  $i$  in Table AT and  $i \geq 2$
6. begin
7.  $AT(i).B = 0$ ;
8.  $AT(i).E = 0$ ;
9.  $AT(i).LC = 0$ ;
10. end
11. pivot=1;
12. repeat
13. begin
14. Choose row  $i$  from Table AT such that  $AT(i).LC$  is maximal among all unmarked rows;
15. if (i==1 or i==pivot)
16. begin
17.  $j = \text{Partition}(R_{AT(i).B}, R_{AT(i).B+1}, \dots, R_{AT(i).E})$ ;
18. Update Table AT and unmark all rows;
19. pivot++;
20. end
21. else /\* Middle partition \*/
22. begin
23.  $j = \text{Partition}(R_{AT(i).B}, R_{AT(i).B+1}, \dots, R_{AT(i).E})$ ;



**Figure 4: The tree representations by using  $T_L$  and  $T_U$ .**



**Figure 5: Illustrations of upward and downward partitions.**

```

24.   if  $(AT(i-1).E - AT(i-1).B) < (j - AT(i).B)$ 
25.   {   Update Table AT and unmark all rows;
26.       pivot++; }
27.   else
28.       { Mark row i;
29.         Merge  $(R_{AT(i).B}, R_{AT(i).B+1}, \dots, R_j)$  and
            $(R_{j+1}, R_{j+2}, \dots, R_{AT(i).E})$  together; }
30.   end
31.   end
32.   until (pivot==K)
end

```

Procedure *Partition* ( $R_i, R_{i+1}, \dots, R_j$ ):

1. Determine  $p^*$  such that  $\delta(p^*) = \max_{p \in \{i, i + \frac{i-i+1}{2} - 1\}} \{\delta(p)\}$ ;
2. Attach nodes  $R_{p^*+1}, R_{p^*+2}, \dots, R_j$  under a new node  $I$  in the tree;
3. Return  $p^*$ ;

To generate the allocation tree, algorithm  $VF^K$  first starts with a configuration where all nodes are attached to the root. Table AT (standing for Auxiliary Table) is created to record the status of the allocation tree as stated from line 1 to line 10 in algorithm  $VF^K$ . Note that since there are three kinds of partitioning, i.e., upward, middle and downward partitions, judiciously applying these partitions is able to reduce the total cost of the allocation tree. To facilitate the description of algorithm  $VF^K$ , we define two subtrees, i.e.,  $T_U$  and  $T_L$ , where  $T_U$  (respectively,  $T_L$ ) is left upper (respectively, right lower) subtree and does not contain the top level (respectively, the last level) of the tree. Consider the tree representations by  $T_U$  and  $T_L$  in Figure 4 where the original tree is the one in Figure 3b. Figure 5a shows the scenario of an upward partition for the allocation tree in Figure 4a. On the other hand, Figure 5b illustrates the scenario of

Data record	$R_1$	$R_2$	$R_3$	$R_4$	$R_5$	$R_6$
$\Pr(R_i)$	0.237	0.211	0.132	0.132	0.08	0.05

Data Record	$R_7$	$R_8$	$R_9$	$R_{10}$	$R_{11}$
$\Pr(R_i)$	0.05	0.027	0.027	0.027	0.027

Table 3. The profile of an illustrative example.

a downward partition for the allocation tree in Figure 4b. From Table AT, algorithm  $VF^K$  chooses the level with the maximal cost to partition (line 14 of algorithm  $VF^K$ ). Let pivot be the number of levels expanded in the allocation tree thus far. Explicitly, as can be seen from line 15 to line 20 in algorithm  $VF^K$ , if the cost of the top (respectively, the last) level is larger than that of other levels, algorithm  $VF^K$  performs upward (respectively, downward) partition. Otherwise,  $VF^K$  performs the middle partition according to the operations from line 21 to line 31 in algorithm  $VF^K$ . Table AT is then updated accordingly. As can be seen from line 24 to line 29 in algorithm  $VF^K$ , algorithm  $VF^K$  makes sure that each partition will satisfy the feature of hierarchy. That is, the number of leaf nodes (i.e., data items) in the upper level is always smaller than that in the lower level. According to Definition 2, the candidate set of nodes with the maximal reduction gain  $\delta(p^*)$  is chosen. When such a set of nodes is identified and moved to the next level, those nodes will be evaluated by themselves to see if any further partition for some of them is necessary in line 32 (the condition ( $pivot == K$ ) means that the number of the channels has been reached). Algorithm  $VF^K$  partitions the nodes iteratively with the objective of minimizing the average cost  $\sum_{i=1}^K w_i * P_{L_i}$  until the depth of the tree reaches the number of broadcast disks in the broadcast disk array. As such, the allocation tree is expanded and constructed in a top down manner.

### 3.2 An Example Execution Scenario of $VF^K$

For example, consider the profile in Table 3 where the number of data items  $n$  is 11 and the number of broadcast disks  $K$  is 4. The initial tree configuration is shown in Figure 6a, where all data records are attached to the root. Procedure *Partition* then determines the optimal partition of nodes to be moved to the next level. The values in Table AT and their changes made in accordance with the execution of algorithm  $VF^K$  are shown in Table 4, where in Table 4a-4c, the first table is Table AT and the second table is the one to determine the cut point  $p^*$  for the partition selected where the maximal value of  $\delta(p)$  is marked with an '\*'. From the calculation shown in Table 4a, we obtain  $p^* = 4$ , and therefore group nodes  $R_5, R_6, \dots$ , and  $R_{11}$  together and move them to the next level, resulting in the configuration shown in Figure 6b. Note that in Table AT, the value of  $AT(i).LC$  is the cost of nodes in level  $i$ , i.e.,  $w_i * P_{L_i}$ , and explicitly, equal to  $C_{AT(i).B}, AT(i).E$ . In Table 4b, since  $AT(1).LC > AT(2).LC$ , the upward partition is performed.

Level i	$AT(i).B$	$AT(i).E$	$AT(i).LC$
1	1	11	5*
2	0	0	0
3	0	0	0
4	0	0	0

p	1	2	3	4	5
$C_{1,11}$	5	5	5	5	5
$C_{1,p} + C_{p+1,11}$	3.4335	2.484	2.05	1.932	2.104
$\delta(p)$	1.5665	2.516	2.95	3.068*	2.896

(a). Partition  $(R_1, R_2, \dots, R_{11})$  is selected and decomposed to  $(R_1, \dots, R_4)$  and  $(R_5, \dots, R_{11})$

Level i	$AT(i).B$	$AT(i).E$	$AT(i).LC$
1	1	4	1.068*
2	5	11	0.864
3	0	0	0
4	0	0	0

p	1	2
$C_{1,4}$	1.068	1.068
$C_{1,p} + C_{p+1,4}$	0.475	0.356
$\delta(p)$	0.593	0.712*

(b). Partition  $(R_1, R_2, \dots, R_4)$  is selected and decomposed to  $(R_1, R_2)$  and  $(R_3, R_4)$

Level i	$AT(i).B$	$AT(i).E$	$AT(i).LC$
1	1	2	0.224
2	3	4	0.132
3	5	11	0.864*
4	0	0	0

p	5	6	7
$C_{5,11}$	0.864	0.864	0.864
$C_{5,p} + C_{p+1,11}$	0.52	0.381	0.342
$\delta(p)$	0.344	0.483	0.522*

(c). Partition  $(R_5, R_6, \dots, R_{11})$  is selected and decomposed to  $(R_5, \dots, R_7)$  and  $(R_8, \dots, R_{11})$

Level i	$AT(i).B$	$AT(i).E$	$AT(i).LC$
1	1	2	0.224
2	3	4	0.132
3	5	7	0.18
4	8	11	0.162

(d). The final result of Table AT.

Table 4. Determining from Table AT the set of nodes to be grouped together as a partition for allocation tree generation.

Following the calculation shown at the right side of Table 4b, we have  $p^* = 2$  and Figure 6b in turn leads to Figure 6c. Similarly, we obtain  $AT(1).LC = 0.224$  and  $AT(2).LC = 0.132$  shown in Table 4c. Since  $AT(3).LC = 0.864$  is the maximal in Table 4c, algorithm  $VF^K$  performs the downward partition. Following the calculation in Table 4c, we have  $p^* = 7$ . Then, the configuration shown in Figure 6d and in Table 4d follows. As the depth of the allocation tree equals the number of broadcast disks, algorithm  $VF^K$  completes. It can be seen that the very advantage of algorithm

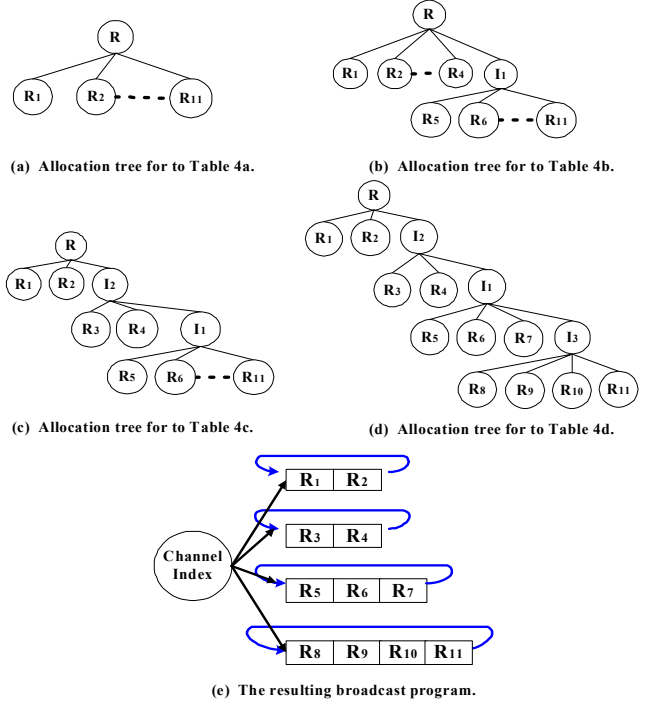


Figure 6: An execution scenario of algorithm  $VF^K$ : (a)-(d) the generation of the allocation tree, and (e) the resulting broadcast program.

$VF^K$  is that even when the number of broadcast channels  $K$  changes dynamically, algorithm  $VF^K$  can reach the new configuration very efficiently with the number of data items required to be moved around broadcast channels minimized. Also, as validated in our experiments in Section 4, when the number of data items to be broadcast is large, algorithm  $VF^K$  can expand the allocation tree very efficiently. According to the configuration in Figure 6d, we can have the hierarchical broadcast program shown in Figure 6e where it can be verified that the average expected delay is the sum of costs of all levels in the tree shown in Figure 6d, i.e.,  $0.224+0.132+0.18+0.162=0.698$ .

## 4. PERFORMANCE EVALUATION

In order to evaluate the performance of algorithm  $VF^K$ , we have implemented a simulation model of the broadcast environment. Specifically, the simulation model is described in Section 4.1. Then, we examine the impact of employing hierarchical broadcast programs in Section 4.2. Performance of algorithm  $VF^K$  is comparatively analyzed in Section 4.3.

### 4.1 Simulation Model

Table 5 summarizes the definitions for some primary simulation parameters. The number of data items to be broadcasted in a broadcast disk array is denoted by  $n$  and the number of broadcast disks in a broadcast disk array is  $K$ . The access frequencies of broadcast data items are modelled

Notation	Definition
n	Total number of data items to be broadcast
K	Number of broadcast disks in a broadcast disk array
$\theta$	Zipf distribution parameter
FLAT	Scheme to generate a flat broadcast program

Table 5. The parameters used in the simulation.

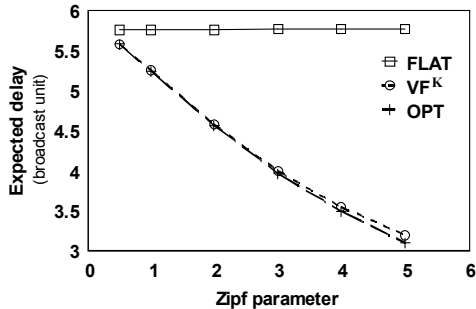


Figure 7: The average expected delay of FLAT, OPT and VF<sup>K</sup> with the value of  $\theta$  varied.

by the Zipf distribution. Let  $P_r(R_i) = (\frac{N-i}{N})^\theta$ , where  $\theta$  is the parameter of Zipf distribution [6]. It can be verified that the access frequencies become increasingly skewed as the value of  $\theta$  increases. For comparison purposes, a scheme that evenly allocates data items to each broadcast disk, referred to as FLAT, is implemented. An example broadcast program generated by FLAT is the one shown in Figure 1a. To obtain the optimal solutions for comparison purposes, we implemented scheme OPT by using the technique of branch and bound. For better readability, the implementation detail of OPT is not included in this paper. Experimental results for algorithm VF<sup>K</sup>, OPT and FLAT have been obtained and comparatively evaluated.

## 4.2 Employing Hierarchical Broadcast Programs

To show the advantage of generating hierarchical broadcast program, we set the value of n to 50 and the value of K to 4. The expected delay of data items under FLAT, OPT and VF<sup>K</sup> are examined with the value of  $\theta$  varied. Without loss of generality, assume that all the data items are of the same size which is used as one unit of waiting time. The resulting expected delay of data items by running FLAT, OPT and VF<sup>K</sup> are shown in Figure 7. It can be seen from Figure 7 that the access frequencies become increasingly skewed as the value of  $\theta$  increases and the difference between expected delay of OPT and that of VF<sup>K</sup> is almost negligible, showing the very high quality of the solutions obtained by algorithm VF<sup>K</sup>. Note that as the access frequencies become increasingly skewed, the hierarchical broadcast programs generated by OPT and VF<sup>K</sup> perform significantly better than the flat broadcast program, indicating the very advantage of exploiting the feature of variant-fanout for the allocation tree generation in algorithm VF<sup>K</sup>.

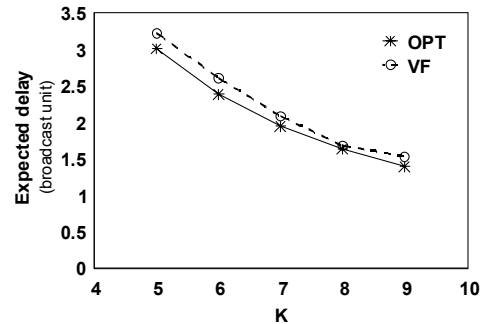


Figure 8: The performance comparison between OPT and VF<sup>K</sup> with the value of K varied.

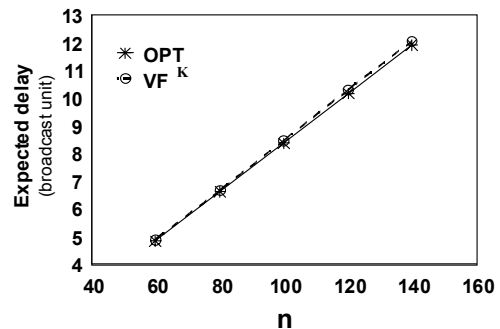


Figure 9: The performance comparison between OPT and VF<sup>K</sup> with the value of n varied.

## 4.3 Comparative Analysis for VF<sup>K</sup> and OPT

We now examine the impact of varying the values of n and K to the performance of VF<sup>K</sup> and OPT. First, in order to evaluate the impact of increasing the value of K, we set the value of n to 50 and the value of  $\theta$  to 3. Figure 8 shows the performance results of OPT and VF<sup>K</sup>.

As can be seen in Figure 8, the expected delay of OPT and VF<sup>K</sup> decreases as the value of K increases. This agrees with our intuition, since as the number of broadcast channels increases, the number of data items in each broadcast channel decreases, thereby reducing the expected delay of data items. Notice that the difference between the expected delay of VF<sup>K</sup> and that of OPT is very small when the value of K varies, again showing the good quality and the robustness of solutions obtained by VF<sup>K</sup>.

Next, the experiments of varying the value of n for OPT and VF<sup>K</sup> are conducted where we set the value of K to 4 and the value of  $\theta$  to 3. Figure 9 shows the performance comparison between OPT and VF<sup>K</sup> when the value of n varies. In Figure 9, as the number of data items to be broadcast increases, the expected delays of data items resulted by OPT and VF<sup>K</sup> increase linearly as we anticipate. Also, the difference between the expected delays resulted by OPT and

$VF^K$  is negligible, again suggesting that algorithm  $VF^K$  be able to find a solution of very high quality efficiently when the value of  $n$  increases.

It is shown by our experimental results that algorithm  $VF^K$  is not only able to produce the solutions of very high quality but also of good scalability which is important for  $VF^K$  to be of practical use to generate hierarchical broadcast programs dynamically in a mobile computing environment.

## 5. CONCLUSIONS

In this paper, we explored the issue of generating hierarchical broadcast programs with the data access frequencies and the number of broadcast disks in a broadcast disk array given. Specifically, we first formulated the problem of generating hierarchical broadcast programs and transformed this problem into the one of constructing a channel allocation tree with variant-fanout. By exploiting the feature of tree generation with variant-fanout, we developed a heuristic algorithm  $VF^K$  to minimize the expected delay of the corresponding broadcast program. Performance of these algorithms was analyzed and sensitivity analysis on several parameters, including the number of data items and the number of broadcast disks, was conducted. It was shown by our simulation results that by exploiting the feature of variant-fanout in constructing the channel allocation tree, the solution obtained by algorithm  $VF^K$  is of very high quality and is in fact very close to the optimal one. By exploring the feature of variant-fanout tree construction, algorithm  $VF^K$  is not only able to produce the solutions of very high quality but also of good scalability which is important for algorithm  $VF^K$  to be of practical use to generate hierarchical broadcast programs dynamically in a mobile computing environment.

## 6. ACKNOWLEDGMENT

The authors are supported in part by the Ministry of Education Project No. 89-E-FA06-2-4-7 and the National Science Council, Project No. NSC 89-2218-E-002-028 and NSC 89-2219-E-002-028, Taiwan, Republic of China.

## 7. REFERENCES

- [1] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik. Broadcast disks: data management for asymmetric communication environments. In *Proceeding of ACM SIGMOD*, pages 199–210, March 1995.
- [2] S. Acharya and S. Muthukrishnan. Scheduling on-demand Broadcasts: New Metrics and Algorithms. In *Proceeding of the 4th ACM/IEEE International Conference on Mobile Computing and Networking*, pages 43–54, October 1998.
- [3] D. Barbara. Mobile Computing and Databases—A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 11(1):108–117, January/February 1999.
- [4] M.-S. Chen, P. S. Yu, and K.-L. Wu. Indexed Sequential Data Broadcasting in Wireless Mobile Computing. In *17th IEEE International Conference on Distributed Computing Systems*, pages 124–131, 1997.
- [5] M. H. Dunham. Mobile Computing and Databases. *Tutorial of International Conference on Data Engineering*, February 1998.
- [6] J. Gray, P. Sundaresan, S. Englert, K. Baclawski, and P. J. Weinberger. Quickly Generating Billion-Record Synthetic Databases. In *Proceeding of ACM SIGMOD*, pages 243–252, March 1994.
- [7] Q. Hu, D. L. Lee, and W.-C. Lee. Performance Evaluation of a Wireless Hierarchical Data Dissemination System. In *Proceedings of the The Fifth Annual International Conference on Mobile Computing and Networking*, pages 163–173, 1999.
- [8] Q. Hu, W.-C. Lee, and D. L. Lee. Indexing Techniques for Wireless Data Broadcast under Data Clustering and Scheduling. In *Proceedings of The Eighth International Conference on Information and Knowledge Management*, pages 351–358, November 1999.
- [9] T. Imielinski, S. Viswanathan, and B. Badrinath. Data on Air: Organization and Access. *IEEE Transactions on Knowledge and Data Engineering*, 9(3):353–372, June 1997.
- [10] J. Jing, A. Helal, and A. Elmagarmid. Client-Server Computing in Mobile Environments. *ACM Computing Surveys*, 31(2):117–157, June 1999.
- [11] W.-C. Lee and D.-L. Lee. Signature Caching Techniques for Information Filtering in Mobile Environments. *ACM Journal of Wireless Networks*, 5(1):57–67, Jan 1999.
- [12] S.-C. Lo and A. L. P. Chen. Optimal Index and Data Allocation in Multiple Broadcast Channels. In *Proceeding of the 16th International Conference on Data Engineering*, pages 293–302, March 2000.
- [13] E. Pitoura and P. K. Chrysanthis. Exploiting Versions for Handling Updates in Broadcast Disks. In *Proceedings of 25th International Conference on Very Large Data Bases*, pages 114–125, September 1999.
- [14] N. Shivakumar and S. Venkatasubramanian. Energy Efficient Indexing for Information Dissemination in Wireless Systems. *ACM Journal of Wireless and Network Application*, 1(4):433–446, January 1996.
- [15] K. Stathatos, N. Roussopoulos, and J. S. Baras. Adaptive Data Broadcast in Hybrid Networks. In *Proceeding of the 23rd International Conference on Very Large Data Bases*, pages 326–335, August 1997.
- [16] C.-J. Su and L. Tassiulas. Joint Broadcast Scheduling and User's Cache Management for Efficient Information Delivery. In *Proceeding of the 4th ACM/IEEE International Conference on Mobile Computing and Networking*, pages 33–42, October 1998.
- [17] WAP application in Nokia. <http://www.nokia.com/corporate/wap/future.html>.
- [18] WAP application in Unwired Planet, Inc. <http://phone.com>.