Regular Paper

An Index Allocation Method for Data Access over Multiple Wireless Broadcast Channels

Damdinsuren Amarmend,[†] Masayoshi Aritsugi[†]
and Yoshinari Kanamori[†]

With the advent of mobile technology, broadcasting data over one or more wireless channels has been considered an excellent way to efficiently disseminate data to a large number of mobile users. In such a wireless broadcast environment, the minimization of both access and tuning times is an important problem because mobile devices usually have limited battery power. In this paper, we propose an index allocation method for data access over multiple channels. Our method first derives external index information from the scheduled data, and then allocates it over the channels, which have shorter broadcast cycles and hotter data items. Moreover, local exponential indexes with different parameters are built within each channel for local data search. Experiments are performed to compare the effectiveness of our approach with an existing approach. The results show that our method outperforms the existing method by 30% in average access time and by 16% in average tuning time when the data skew is high and data item size is much bigger than index node size.

1. Introduction

The advanced achievements in wireless network and portable computer technologies have made it a reality that people access their desired data at anytime from anywhere using their mobile computing devices. In general, wireless environments have the following properties:

- Mobile devices have limited capacities in power,
- Mobile devices have much weaker communications capacities than fixed host computers do,
- Large volume of mobile users will access the shared data, and
- Most users are expected to be highly mobile.

Periodic data broadcasting compensates these aspects, and is considered to be an excellent solution to disseminate data in wireless environments.

A user can access the broadcast data just by tuning into the channel. However, data can only be accessed in sequential order. The users need to wait until the data of interest appear on the broadcast channel. How to improve the efficiency of data broadcasting becomes a challenging issue [7),10)].

To analyze the performance of the broadcasting methods, two metrics have to be considered: access time and tuning time. Access time is measured from the moment when a user initially poses a query to the moment all the result is acquired. Tuning time is the amount of time a user actively listens to the broadcast channel. By reducing these times, the related work aim to contribute to reduction of power consumption of mobile users and shortening the delay required for getting their information from wireless channel. The actual power consumption of mobile devices is essentially reflected by the tuning time reduction. In general, it is difficult to say that how much power is saved by reducing these times. For example, it may depend on what kind of chipset is used. In Ref. 18), it is given that the Hobbit chip from AT&T consumes 5,000 times less power when it operates in doze mode ($50\,\mu W$) than in active mode ($250\,mW$). Let us take the Hobbit chip for example and compute how much power would be saved by reducing the times. If we assume that the device spend T time for actively listening to the channel and I time for dozing off for fulfilling a request, then the power $P = 250000\,\mu W \times T + 50\,\mu W \times I$. Now, let us suppose that we have saved the access time by 30% and the tuning time by 20%. Then, the saved power will be as follows: $P = 250000\,\mu W \times 0.2 \times T + 50\,\mu W \times I \times 0.3 = 5000\,\mu W \times +15\,\mu W \times I$. Our work and the related work aim to optimize this issue from the server side.

To improve the average access time, many data scheduling methods have been in-

---

† Faculty of Engineering, Gunma University

vented [1),4),6),13),14),16),20)].

Moreover, many research efforts have been reported in the literature that aim at the improvement of data broadcast efficiency by providing index information [8),11),18),21)]. Indexes let the mobile users stay in the doze mode most of the time and only come to the active mode when the data of interest appear on the channel. However, mixing indexes with broadcast data increases broadcast cycle length. As a result, the average access time increases. Therefore, an optimal allocation of indexes to the broadcast channels is an important issue.

In this paper, we contribute an index allocation method for the broadcast data with different access frequencies over multichannels. Our method minimizes the average access and tuning times with the help of the external index and the optimal allocation of them to the broadcast channels. The extended abstract of the contents of this paper was presented at Ref. 5). The rest of the paper is organized as follows. Section 2 introduces the related work on the problem. In Section 3, the proposed index method is explained in detail. Then, we analyze the performance of our method in Section 4. Finally, Section 5 presents conclusions and future work.

2. Related Work

2.1 Data Broadcast Methods

Acharya et al. propose Broadcast Disks, in which hot (more frequently accessed) data items are allocated more frequently than cold (less frequently accessed) data items in each broadcast period [1),4)]. By shortening the broadcast cycles of hot data, the average access time decreases. Their work is improved further in Refs. 2), 3), 6). Vaidya and Hameed considers optimal broadcasting for variable-sized data in Ref. 20). In Ref. 13), the authors considered broadcasting semantically related data. The above methods assumed only a single broadcast channel. The data broadcasting over multiple broadcast channels, which can not be coalesced in to a single physical channel, is considered in Refs. 14), 16).

2.2 Index Methods for Data Broadcast

Imielinski et al. propose three indexing techniques for accessing data items on the broadcast channel. (1,m) indexing in Ref. 18) simply replicates the whole index tree m times in each broadcast cycle. Distributed indexing [18)] remedies it by replicating only an upper part of the tree. The third technique called flexible indexing [11)] is non-tree structured method. In flexible indexing, the broadcast data is divided into several equi-sized segments. Each segment contains some binary-based index information of data in other segments. Their work proves that providing index with broadcast data saves a significant amount of energy of mobile devices.

Xu, et al. extend the idea of flexible indexing and propose exponential index [21)]. The difference from flexible indexing is that any index base can be used. By adjusting the two parameters (the number of segments and the index base), it improves the performance. By their performance analysis, exponential index outperforms flexible indexing and distributed indexing. The above indexing methods assume that all the data items have the same access frequencies. Taking access frequency into account is important to improve the efficiency of indexing methods. Chen et al. explored the issue of indexing data with skewed access. To minimize the average tuning time, they considered unbalanced index trees with fixed and variant fanout [8)]. However, these work assumed only flat broadcast program.

Tan and Yu [19)] propose an indexing method for data broadcast with skewed access. One thing about all these work is that they assume data is broadcast over a single broadcast channel.

Recently, some research work discuss the problem of minimizing the access and tuning times in multiple channels environments [9),12),15)~17),22)]. In Ref. 16), some efficient algorithms for broadcasting data to mobile users over multiple channels are proposed. The index allocation method they employed is inefficient, since the whole (1,1) index is replicated once in a broadcast cycle in each channel. Two approaches are observed among these methods. One approach interleaves indexes with data over all channels [9),15),22)]. In contrast, in the other approach, data and index are broadcast in separate channels [12),17)].

Hsu, et al. extend distributed indexing into multiple channel environments. In this method, hot data and its index tree nodes are replicated several times in one broadcast cycle [9)]. Yee and Navathe propose some techniques of stripping index nodes over multichannels [20)]. Their methods are based on balanced index trees, which do not capture the different access frequencies of data nodes. Lo and Chen proposed

topological tree, in which data and index are mixed and broadcast in the same time slot [15]. Topological tree is based on an unbalanced tree to reflect the different access frequencies of data nodes. Their approach consider the sizes of index and data nodes are the same. In practice, index nodes are much smaller than data nodes.

In contrast to these approaches, Shivakumar and Venkatasubramanian propose k-ary Alphabetic Huffman Tree and allocate it to the dedicated index channel [17]. However, this approach is not efficient when the number of index channels is less than the depth of the index tree. In Ref. 12), Jung et al. propose a tree-structured index allocation method, which is based on Alphabetic Huffman Tree. They allocate the index tree to the dedicated index channels with the same frequencies as the related data. All these work for multichannel environments have tree structures. A user has to wait for the root node to access any data. To decrease this waiting time, above methods replicate the upper parts of the index tree many times within a broadcast cycle. As a result, the broadcast cycle becomes longer increasing the average access time.

Unlike these approaches, our method has the following properties at the same time:
- Non-tree structured but well distributed.
- The given channels are not separated.

3. Index Allocation over Multiple Broadcast Channels

In our method, we do not separate the channels in order to fully utilize the channel bandwidths. We have the following assumptions throughout this paper.
- All the broadcast channels are assumed to have the same bandwidth.
- Given k broadcast channels, we number them 1 through k. For the broadcast channels, upper means their channel number is closer to 1, lower means closer to k.
- The switching time from one channel to another is considered to be negligible.
- The sizes of all the data nodes are assumed to be the same. And the sizes of the index nodes are assumed to be the same. However, the size of a data node can be larger than that of an index node.
- An index node is assumed to have the information about the channel number and the time offset for the next index node or data nodes.

- A data (or index) broadcast cycle is one period of broadcast data (or index).

Our method consists of two parts: external index and local indexes. A user first retrieves the channel number of data of interest from the external index. After that, the user searches in that channel using local indexes.

External index can be derived from any scheduled data. External index entries are packed into nodes, and then allocated to the upper channels. This allocation is based on the following observations. First, the size of the external index is relatively much smaller than the size of the broadcast data. Secondly, data broadcast algorithms for multichannel environments such as Refs. 12), 16) tend to allocate a few hot data to the upper channels, and more and colder data to the lower channels. If the data node size gets bigger, the difference between broadcast cycle lengths of the top and the bottom channels increases proportionally. Therefore, allocating the external index nodes on upper channels will not affect much to the average access time.

For local searches within a channel, in principle, any index method for a single channel can be used. To make the discussion in this paper concrete, we employ exponential index [21] for this purpose.

In the next sections, we discuss how to build external and local indexes and users access algorithm.

3.1 Data Allocation over Multiple Channels

Our proposed index can be based on any good data broadcast algorithm for multiple channels environments. In this paper, we adopt the data broadcast methods used in Refs. 12), 16). This method assumes every data item has a temperature, which equally means an access frequency. In this paper, we take temperature as access probability.

At first, the data set is sorted according to their temperatures. The average channel temperature is found by dividing the sum of all the temperatures to the number of channels, say k. Starting from the hottest data item to the coldest one, the algorithm allocates a data item to a channel if the sum of that channels' temperatures is less than the average channel temperature.

We also adopt an example of this broadcast program from Ref. 12), which is illustrated in **Fig. 1**. In the example, 14 data are allocated

Fig. 1 An example of data broadcast program.

according to their temperatures to 4 channels. Among the others, $[E, N]$ have the highest temperatures while $[I, J, F, G, H]$ have the lowest temperatures.

3.2 The External Index

The purpose of the external index is to let users quickly find out in which channel their data of interest is being broadcast. In this section, we explain how to derive it from an existing data broadcast program.

Definition 1 Let us have given k number of channels, and N number of data items. We define a working set (WS), which includes all the data items in ascending order by their index keys.

Definition 2 Let N_i be the number of data allocated to channel i. Let $min(WS)$ denote the element of WS with the minimum index key. Similarly, let $max(WS)$ denote the element with the maximum index key. A data list D_i contains all the data in channel i in ascending order. We add $min(WS)$ and $max(WS)$ if they are not included in D_i. Thus, $D_i = \{min(WS), d_1, \ldots, d_{N_i}, max(WS)\}$, where $min(WS) \le d_1 \le \ldots \le d_{N_i} \le max(WS)$.

Definition 3 A set I_j can be derived from WS for every sequential two data items, d_j and d_{j+1}, in D_i. For any $e \in I_j$ except $min(WS)$ and $max(WS)$, it holds that $d_j < e < d_{j+1}$. If $min(WS)$ and $max(WS)$ are not originally in D_i, then they will be included in I_j. Also, the data items on the upper channels will not be entered in I_j. From I_j, we define a triple of values $[min(I_j), max(I_j), ch(I_j)]$ as an external index entry. The first and second are minimum and maximum elements of I_j, respectively. The third one is the highest channel number of elements of I_j. The elements of I_j are allocated to the different channels. In other words, $ch(I_j)$ finds the location of the data with the highest access frequencies in I_j.

With these definitions, all the index entries must be derived for the channels 1 through $k-1$, except channel k. For the data in the last channel, we do not need to derive index entries from them, since those data are already indexed

1. $WS = \{d_1, d_2, \ldots, d_N\}$: the working set
2. N_i: number of data items allocated to channel i
3. $SD = \{D_1, D_2, \ldots, D_k\}$ the set of scheduled data over k channels, where
4. $D_i = \{d_{i1}, d_{i2}, \ldots, d_{i_{N_i}}\}$
5. Sort WS in ascending order;
6. Sort each D_i in ascending order;
7. for $(i = 1; i \le k-1; i++)$
8.   if $(d_{i_1} <> min(WS))$ then
9.   Insert $min(WS)$ into D_i;
10. $N_i = N_i + 1$
11.   end if
12.   if $(d_{i_{N_i}} <> max(WS))$ then
13.   Insert $max(WS)$ into D_i
14. $N_i = N_i + 1$
15.   end if
16.   for $(j = 1; j < N_i; j++)$
17.   Determine I_j for $[d_{i,j}, d_{i,j+1}]$ from WS
18.   Determine and save a triple
      $T = [min(I_j), max(I_j), ch(I_j)]$
      by Definition 3
19.   end for
20. end for
21. Remove D_i's original elements from WS
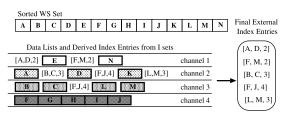22. Remove any duplicates from the external index set

Fig. 2 An example of external index entries.

in the upper channels. The derivation method is illustrated in Algorithm 1.

In the running example, the working set is initially $WS = \{A, B, C, D, E, F, G, H, I, J, K, L, M, N\}$. The result of the Algorithm 1 on this example is shown in **Fig. 2**. On the left side, the entries derived from the corresponding I sets are shown. The first index entry $[A, D, 2]$ is obtained from the set $I_1 = \{A, B, C, D\}$ which corresponds to E. Among the elements of I_1, A and D have the highest access frequencies. So, $ch(I_1)$ is set to 2. The final set of the derived index entries contains two $[F, J, 4]$ entries. The first one is from the interval $[D, K]$, and the second is from $[C, L]$. This because $[C, L]$ includes $[D, K]$ in WS. By removing the last one, the final set is displayed on the right side of the figure.

3.3 Allocation of External Index

A user finds out that his data of interest is allocated to channel i if the intervals of the exter-

nal index entries, which belong to that channel, do not include the key of the data. The average access time for a request consists of two times: $T_{ext.acc}$ - average access time for traversing external index, and $T_{loc.acc}$ - time for searching in local index. Similarly, the average tuning time for a request consists of two parts: $T_{ext.tun}$ - average tuning time for the external index traversal, and $T_{loc.tun}$ - average tuning time for the local index traversal plus the data retrieval.

The total expected average access time for a request can be expressed as:

$$T_{acc} = T_{ext.acc} + T_{loc.acc}, \tag{1}$$

and the total expected average tuning time for a request as:

$$T_{tun} = T_{ext.tun} + T_{loc.tun}. \tag{2}$$

From these equations, in order to minimize T_{acc} and T_{tun}, we have to minimize the component costs: $T_{ext.acc}$, $T_{loc.acc}$, $T_{ext.tun}$, and $T_{loc.tun}$. In this section, we attempt to minimize $T_{ext.acc}$ and $T_{ext.tun}$ by allocating the external indexes optimally.

We accommodate the entries into external index nodes. We consider two approaches for the allocation of external indexes over multichannels.

The first approach is to allocate the index nodes according to which channels they have been derived from. In each channel, the external index nodes will be located contiguously.

Let B_i and N_i be the number of external index nodes and the number of data assigned to channel i, respectively. Here, $Temp_i$ denotes the sum of temperatures of data items in channel i while $temp_{ij}$ denotes the sum of temperatures of data items which are pointed by the jth index node of the channel i. Moreover, R is the ratio of an index node size to a data node size. We measure the costs in terms of index node. Due to the non-uniform data allocation algorithm, the number of data in the channels will be sorted as $N_1 \leq N_2 \leq \ldots \leq N_{k-1}$. The broadcast cycle length of channel i will be $B_i + N_i \times R$. When R and the number of data are high but the given number of channels is relatively low, it is highly expected that the cycle lengths of the channels will be sorted in ascending order as $B_i + N_i \times R \leq B_j + N_j \times R$, where $i \leq j$. The external indexs nodes will be assigned to channels 1 through n, where $1 \leq n \leq k - 1$. Any search will be started by finding the first external index node in channel 1. If the desired item is not included in the external index entries, then that item must be

broadcast on the channel. If the item is in another channel, the search will be jumped to that channel. The expected average access time for traversing the external index is as follows:

$$
\begin{aligned}
T_{ext.acc} = & \\
& \sum_{i=1}^{n} \Bigg(\frac{(B_i + N_i \times R)}{2} \\
& \times \Bigg(\sum_{j=1}^{B_i} temp_{ij} + Temp_i \Bigg) \\
& + \sum_{j=1}^{B_i} j \times temp_{ij} + B_i \times Temp_i \Bigg)
\end{aligned} \tag{3}
$$

The first term of Eq. (3) indicates the average time for waiting for the beginning of the external index in channel i. The second term is the time for traversing external index for the data items, which are broadcast on other channels. The third one shows the average external index traversal time for the data items, which are broadcast on channel i.

And, the expected average tuning time for traversing the external index can be found as follows:

$$
\begin{aligned}
T_{ext.tun} = \sum_{i=1}^{n} \Bigg(\frac{1+R}{2} + \sum_{j=1}^{B_i} j \times temp_{ij} \\
+ B_i \times Temp_i \Bigg).
\end{aligned} \tag{4}
$$

In each channel, any node will contain an offset to the beginning of the external index in that channel. Thus, before reading the external index in channel i, we read one index or data node. This explains the first term in Eq. (4). The second and third terms of Eq. (4) are explained similarly with Eq. (3).

From the above Eqs. (3) and (4), it can be observed that access and tuning times will be very high when R and the number of data are high but the given number of channels is relatively lower. In that case, the broadcast cycle lengths of the lower channels are much longer than the lengths of broadcast cycles on the upper channels. Therefore, users will wait for a long time to read the external index in the lower channels. For this reason, this allocation is not considered favorable.

The other method in our consideration allocates the nodes to the channels more flexibly yet efficiently. In this method, more nodes are allocated to the upper channels than to the lower channels to avoid the long waits in

the lower channels. In other words, it will be $B_1 \geq B_2 \geq \ldots \geq B_m$, where B_i is the number of index nodes allocated to channel i. As a result, the external index nodes are allocated to fewer channels 1 through m, where $1 \leq m \leq n$, than to the first method. In the extreme case, we put all the external index nodes on channel 1. Therefore, the first method becomes a case of the second.

To indicate the relation of the index entries to the belonging channel, we should add one more attribute to the index entry. So, the external index entry for the second method becomes as $[min, max, ch1, ch2]$, where $ch1$ is for the next channel number to traverse, and $ch2$ for which channel this entry belongs to.

The expected average access time for the index traversal becomes

$$T_{ext.acc} =$$
$$\sum_{i=1}^{m} \left(\frac{(B_i + N_i \times R)}{2} \right.$$
$$\times \left(\sum_{j=1}^{B_i} temp_{ij} + Temp_i \right)$$
$$+ \left. \sum_{j=1}^{B_i} j \times temp_{ij} + B_i \times Temp_i \right), \quad (5)$$

and the expected average tuning time is:

$$T_{ext.tun} = \sum_{i=1}^{m} \left(\frac{1+R}{2} + \sum_{j=1}^{B_i} j \times temp_{ij} \right.$$
$$\left. + B_i \times Temp_i \right). \quad (6)$$

Equations (5) and (6) are similar to Eqs. (3) and (4) except for m, n, and B_i variables. In general, we expect B_i is much less than N_i, especially when the size ratio R is high. Therefore, putting the external index nodes on upper channels with short broadcast cycle lengths, saves us from waiting for a long time in the lower channels. Therefore, we employ the latter method in this paper.

For the example in Fig. 2, let the data items have the temperatures as shown in **Table 1**. The first and the second column show the channel number and its data items. The last one displays the sum of temperatures of the data items. The data items in the same channel are considered to have the same temperatures.

Using this configuration, in **Table 2** we calculated the expected average access times for different external index allocations and differ-

Table 1 Temperatures for the data items in the example in Fig. 2.

Ch	Data Items	Sum of Temperatures
1	E, N	$0.4 = 0.2 \times 2$
2	A, D, K	$0.3 = 0.1 \times 3$
3	B, C, L, M	$0.2 = 0.05 \times 4$
4	F, G, H, I, J	$0.1 = 0.02 \times 5$

Table 2 The expected average access times for the Example in Fig. 2.

Size Ratio	Ext. Index Allocation			
	$(2:3)$	$(3:2)$	$(4:1)$	$(5:0)$
$R = 1$	9.35	10.7	11.05	10.1
$R = 10$	35.45	38.7	37.75	32.5
$R = 30$	101.95	97.7	93.75	76.6
$R = 50$	175.95	168.7	161.75	132.6

Algorithm 2 External Index Node Allocation

1. Derive an initial allocation by the first method
2. $N_B = \sum_{i=1}^{n} B_i$
3. Compute this allocation's average access time by Equation (4) and assign it to $minAcc$
4. $for(i = B_1; i \leq N_B; i = i + 1)$
5.   Find all the partitions B_2 through B_m satisfying $B_1 \geq B_2 \geq ... \geq B_m$ and $1 \leq m \leq k$.
6.   For (each such partition) Compute the average access time by the Equation (5) and assign it to $tAcc$
8. $if(tAcc < minAcc)$
9. $minAcc = tAcc$; save the partition
10. $end\ for$
11. Allocate the external index nodes according to the saved partition with $minAcc$

---

ent R's values. The $T_{loc.acc}$ is computed by the following formula considering that there are no local indexes:

$$T_{loc.acc} = \sum_{i=1}^{n} \frac{(B_i + N_i \times R)}{2} \times Temp_i. \quad (7)$$

The $(2:3)$ means the first two index nodes are allocated to channel 1 whereas the last three index nodes are assigned to channel 2. The $(2:3)$ allocation is by the first method, in which $[A, D, 2]$ and $[F, M, 2]$ are on the channel 1, and $[B, C, 3]$, $[F, J, 4]$, and $[L, M, 3]$ are allocated to the channel 2. The other three are the variations of the second method. From the results, we see that the last three allocations outperform the $(2:3)$ when R increases. It shows that we should put more index nodes on the upper channels when R increases.

We devise a greedy algorithm for the optimal allocation of the second method in Algorithm 2. After allocation of the external index nodes, we can duplicate the index nodes within channels to decrease the waiting time. However, this
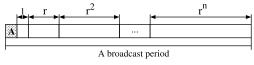
Fig. 3   Division of the next chunks.

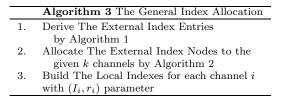issue will not be covered in this paper.

3.4 Building Local Indexes

For the local search in a channel, we extend the exponential index method [21]. Originally, the exponential index is proposed for uniform data access over a single channel data broadcasting. They considered a bucket as a logical broadcast unit, and all the formula and algorithms were based on the bucket conception. However, we take an index node as a broadcast unit. External and local indexes have the nodes of the same size. The size of data node can be different from the index node size. Therefore, we explain our extension in terms of nodes in this paper.

In this method, the broadcast data is divided into chunks. Each chunk has I (called chunk size) data nodes, and an index information of the next chunks. The index consists of two parts: global and local. The local part plainly contains the keys of I data nodes within the chunk. If the key in search is found to be the ith entry in the local part, it means that the desired data node will come after $i - 1$ data nodes.

The global part indexes the next chunks. For a chunk, the global index splits the next chunks into parts with the following lengths: $1, r, r^2, \ldots, r^n$ as in **Fig. 3**. Here, r is called an index base.

The first entry of the global index describes the maximum key value of the next $r^0 = 1$ chunk. The next entry is the maximum key value of the next r chunks, and so on. In other words, ith entry is the maximum key value of a segment of r^{i-1} chunks(i.e., the chunks that are $\lfloor \sum_{j=1}^{i-2} r^j + 1 \rfloor = \lfloor \frac{r^{i-1}-1}{r-1} + 1 \rfloor$ to $\lfloor \sum_{j=1}^{i-1} r^j \rfloor = \lfloor \frac{r^i-1}{r-1} \rfloor$ away). If we consider R and I, the first node of the segment pointed by the ith entry will come after $\lfloor \frac{r^{i-1}-1}{r-1} + 1 \rfloor \cdot (I \cdot R + 1) + I \cdot R$ broadcast units, since each chunk has I number of data nodes plus an index node, including the local chunk. This method indexes not only the current broadcast period but also the next broadcast period. Its performance can be improved and adjusted by tuning I and r parameters. Exponential index is not tree struc-

---

Algorithm 3 The General Index Allocation

1.   Derive The External Index Entries
     by Algorithm 1
2.   Allocate The External Index Nodes to the
     given k channels by Algorithm 2
3.   Build The Local Indexes for each channel i
     with (I_i, r_i) parameter

Fig. 4   An example of the broadcast with the external and local indexes.

tured but has advantages over tree structured methods. It is possible to start searching from any chunk index, therefore naturally better distributed in the wireless broadcast environment than tree structured methods.

We employ this method for our local index in each channel with different parameters. The parameter values will depend on the broadcast cycle lengths of the channels. The local indexes will be allocated in the start of the related data chunks. Given k broadcast channels, we have $(I_1, r_1), (I_2, r_2), \ldots, (I_k, r_k)$ parameters to tune. In order to achieve a good performance, we have to find optimal values for these parameters. The general index allocation algorithm is illustrated in Algorithm 3.

An example of our broadcast method with the external and local indexes is shown in **Fig. 4**. The external index nodes on channel 1 have shorter broadcast cycles than the external index nodes on channel 2.

3.5 User Access Algorithm

In this section, we discuss how a user accesses the data of interest over the multiple channels. Each access starts by reading the first node of the external index in channel 1. To find the first external index node in a channel, a user needs to know the following information:

- $Bcast(i)$: the broadcast cycle length of channel i in terms of index nodes
- $ExtOFF(i)$: the offset of the first node of external index in channel i

We set $ExtOFF(i) = 1$ in this paper, because we put the external index nodes in the head of the broadcast in each channel. Algorithm 4 shows the method of finding the position of the first external index node in channel i.

The access method for this part is illustrated

Algorithm 4 Compute position of Ext.Index
---

1. $/ * channel = i * /$
2. $period = \lfloor current\_time / Bcast(channel) \rfloor$
3. $position = period \times Bcast(i) + ExtOFF(i)$
4. $while(position < current\_time)$
5. $\quad period + +;$
6. $\quad position = period \times Bcast(i) + ExtOFF(i)$
7. $end\ while$

Algorithm 5 Search for External Index
---

1. $ch = 1$
2. Compute *position* of the first ext. index node in channel *ch* by the Algorithm 4
3. $while(current\_time < position)$
4. $\quad$ Wait in the doze mode.
5. for (each ext. index entry e) do
6. $\quad if$ ($e.ch2 <> ch$) then goto 10
7. $\quad if$ ($e.min \leq Key \leq e.max$) $then$
8. $\quad\quad ch = e.ch1$; goto 2
9. $\quad end\ for$
10. Search through the Local Indexes in channel *ch* by the Algorithm 6

Algorithm 6 Search for the Local Index
---

1. Wait until the first local index node of the next chunk to appear
2. Check the local index
3. if (the data item is within the key range specified by the ith entry) $then$
4. $\quad$ Go into the doze mode, wait for the ith node,
5. $\quad$ retrieve the data, and the query is finished
6. $end\ if$
7. Check the global index
8. if (the data item is within the key range specified by the ith entry) $then$
9. $\quad dist = \lfloor \frac{r^{i-1}-1}{r-1} + 1 \rfloor \cdot (I \cdot R + 1) + I \cdot R$
10. $\quad if$ (external indexes within the interval)
11. $\quad\quad then\ dist = dist + B_i\ end\ if$
12. $\quad$ Go into the doze mode, wait for $dist$ broadcast units, goto 2
13. $end\ if$

in Algorithm 5. If the data of interest is not included in the intervals of the index entries for the searching channel, it is known that the data is broadcast in that channel. Otherwise, the next channel number to search in must be found from the index entries related with the current channel. The local search method is shown in detail in Algorithm 6.

4. Performance Analysis

For convenience, we name our method as AIM (for Air Index for data access over Multiple broadcast channels). In this section, we analyze the performance of our method in comparison with a tree structured index allocation method SIRAH (Separated Index Replication based on Alphabetic Huffman tree) proposed in

Ref. 12). In Ref. 12), SIRAH is compared with other similar methods TOPO [15] and ALH [17]. The authors [12] conclude that SIRAH outperforms TOPO and ALH in almost all the configurations of their experiments. Therefore, it is enough for our method, AIM, to compare its performance only with SIRAH. SIRAH separates the channels into two parts: index channel and data channel. They suggested a method to find the optimal ratio of this separation. The method computes average access and tuning times for all the ratios for a given environment. The ratio that gives the minimum average access and tuning time becomes the optimal one. The computation takes the sizes of data and index nodes, number of data, and *NRF*, *IL* (parameters which are derived from SIRAH index allocation algorithm). In their environment, $2 : 8 = index : data$ for total of 10 channels was the optimal ratio. When data node size is big and number of data is high, it is very intuitive that much more channels must be allocated to data than index. We adopted all the parameters and the data distribution from Ref. 12) to create the same experimental environment for fair comparisons. The index and data are allocated by the same algorithm of Ref. 12). Therefore, we adopted the $2 : 8$ ratio as optimal in our experiments.

SIRAH allocates index nodes to the index channels by combining Alphabetic Huffman Tree (AHTree) [17] with a data broadcast schedule. They consider that the data items allocated to the upper channels are closer to the root of the AHTree than the data items on the lower channels. Also, the ancestor nodes of the data items on the upper channels are expected to have higher frequencies in a broadcast cycle than other index nodes. The index nodes of AHTree closer to the root node are allocated first, and with high frequencies. Index nodes with immediate parental relationships must be allocated to one channel, otherwise they can be allocated to separate channels.

4.1 Simulation Model and Workload

The simulation was carried out on a Pentium IV 1.4 GHz with 754 MB memory running Red Hat Linux version 9. We implemented the simulator in standard C language. The nonuniform (or skewed) access pattern of mobile users is modeled by the Zipf distribution with θ parameter [23]. This distribution models the nonuniform access by tuning the θ parameters value between 0 through 1. For example, a highly

Table 3 Simulation parameters.

Parameter	Value
Total number of channels	10
channels for AIM	10
index channels for SIRAH	2
data channels for SIRAH	8
Number of broadcast data	3,000
Node size ratio R	1–50
θ	0.95
Number of requests	5,600
The size of an index node	1,280 B

skewed access is modeled when θ is closer to 1. Since the setting $\theta = 0.95$ is common in broadcast environments [12),16)], we take this value as default. The other parameters used in the simulation are listed in **Table 3**. To make the comparison of the performances of the SIRAH and AIM fair, we used the same size of index node (1,280 B) [16)].

We chose the average access and average tuning times as our primary performance metrics. We built different request sets with nonuniform access pattern for various number of broadcast data. The average access and tuning times are obtained by averaging the access and tuning times of those requests. All the metrics are measured in terms of broadcast units. A broadcast unit is a time to broadcast an index node. Downloading a data node will require R broadcast units.

4.2 Effects of the Ratio a Data Node Size to an Index Node Size

In this section, we discuss how the changes in the ratio of a data node size to an index node size affects the performance of AIM and SIRAH. When the node size ratio R increases, the broadcast cycle lengths become longer. Thus, the average access time of all the data naturally increases. The results are shown in **Fig. 5** and **Fig. 6**. From Fig. 5, our approach significantly outperforms (by 30% when $R = 50$) SIRAH for the average access time. The difference of the performance tends to be bigger when R increases. The reason is that our approach better utilizes the bandwidths of given k channels than SIRAH by not separating the channels. Also, the optimal allocation of external index nodes greatly helps to minimize the average access time. The average tuning time comparison is shown in Fig. 6. AIM performs 16% better than SIRAH when $R = 50$.

4.3 Effects of the Number of Broadcast Data
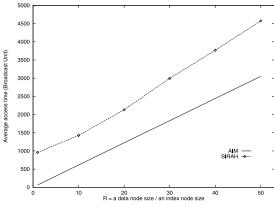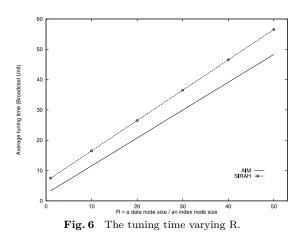
In this section, we investigate how increas-

Fig. 5 The average access time varying R.

Fig. 6 The tuning time varying R.

ing the number of broadcast data affects the performances of AIM and SIRAH. In this experiment, the node size ratio R is set to 50. The number of data to broadcast is increased from 500 to 5,000 with the step of 500. **Fig. 7** and **Fig. 8** show that AIM has much better performance than SIRAH on all the cases. Note that the scale values of vertical axes of these graphs are not started from 0. The lengths of broadcast cycles increase when the number of data increases. Thus the average access times of both methods increase. However, the average access time of AIM increases very slowly. When the number of broadcast data increases, the AHTree becomes bigger. That means the broadcast cycle lengths get longer in the dedicated index channels. This explains the sharp increase of the graph for SIRAH. AIM outperforms by 27% when number of data items reaches 5,000.

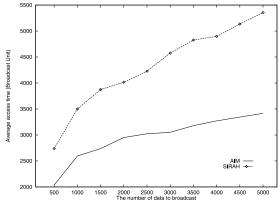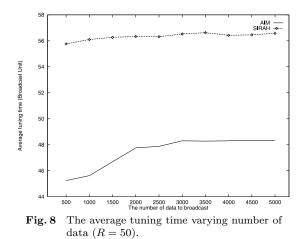Figure 8 also shows the average tuning times of the two methods. Although the tuning times

Fig. 7   The average access time varying number of data ($R = 50$).

Fig. 9   The average access time varying θ ($R = 50$).

Fig. 8   The average tuning time varying number of data ($R = 50$).

Fig. 10   The average tuning time varying θ ($R = 50$).

of both methods looks stable, we observe that AIM outperforms by 19% when number of data items is less than or equal to 1,000, and by 14% from that point up.

4.4   Effects of Various Access Patterns

In this section, we study the behaviour of our index allocation method AIM when user access patterns change. In this experiments, the access frequencies are changed according to the Zipf parameter θ. According to the Zipf distribution, as θ tends to 1, users access patterns are very skewed. That is, few data items have high access frequencies. For our method, the performance should be low when θ tends to 0 (or the access pattern gets closer to uniform distribution). The waiting times of external index nodes in the lower channels affect this. When the access pattern gets less skewed, the performance of SIRAH must become worse. Because users should spend a long time for retrieving cold index nodes from AHTree. These observations are proved by the results in **Fig. 9** and
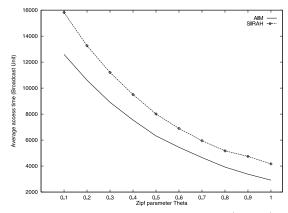
Fig. 10. Note that the scale values of vertical axes of these graphs are not started from 0. From the graphs, we see that the average access times of the two methods decreases when the value of θ tends to 1. However, AIM still performs better at each points than SIRAH. The difference decreases from 21% at $\theta = 0.1$ to 16% at $\theta = 1$. The tuning time for AIM sharply drops by about 20% when θ reaches 1. This is because first, fewer data nodes are allocated to the upper channels by AIM than by SIRAH, secondly, the traversing external indexes on the upper channels incurs less tuning times for retrieving hot data.

5.   Conclusion

In this paper, we proposed an efficient yet flexible index allocation method for broadcasting data with nonuniform access patterns over multiple wireless channels in mobile environments. The existing methods in this domain mainly have tree structures, thus they need to

replicate the tree nodes. Some of them separates the channels into dedicated index and data channels. These replication and separation make the broadcast cycles longer.

Therefore, due to the broadcasts sequential nature, the access time is long in these approaches. Unlike these approaches, our method minimizes average access and tuning times with the help of external index. The optimal allocation of external index nodes improves the performance. We employ exponential index for local data searching within a channel. We extend this method by building exponential indexes with different parameters for each channel. Tuning these parameters, we are able to improve the performance of our method, too.

We analyzed the performance of our method in comparison with the existing method SIRAH in various experiments. The results of the simulated experiments prove that our approach performs much better than SIRAH. However, in case of enough number of available channels, SIRAH may perform better than AIM. In future work, we will think two issues: consideration of our method in more realistic error-prone environments and the actual effect of the access and tuning times in power consumption of mobile devices.

References

1) Acharya, S., Franklin, M. and Zdonik, S.: Dissemination-based Data Delivery Using Broadcast Disks, *IEEE Personal Communications*, Vol.2, No.6, pp.50–61 (1995).
2) Acharya, S., Franklin, M. and Zdonik, S.: Disseminating Updates on Broadcast Disks, *Proc. VLDB Conf.*, pp.354–365 (1996).
3) Acharya, S., Franklin, M. and Zdonik, S.: Prefetching from a Broadcast Disk, *Proc. IEEE Int'l Conf. on Data Engineering*, pp.276–285 (1996).
4) Acharya, S., Franklin, M., Zdonik, S. and Alonso, R.: Broadcast Disks: Data Management for Asymmetric Communication Environment, *Proc. ACM SIGMOD Int'l Conf. Management of Data*, pp.199–210 (1995).
5) Amarmend, D., Aritsugi, M. and Kanamori, Y.: An Air Index for Data Access over Multiple Wireless Broadcast Channels, *Proc. 22nd Int'l Conf. on Data Engineering (ICDE 2006)*, p.135 (2006).

6) Bar-Noy, A., Patt-Shamir, B. and Ziper, I.: Broadcast Disks with Polynomial Cost Functions, *Wireless Networks*, Vol.10, No.2, pp.157–168 (2004).
7) Barbara, D.: Mobile Computing and Databases—A Survey, *IEEE Trans. on Knowledge and Data Engineering*, Vol.11, No.1, pp.108–117 (1999).
8) Chen, M., Yu, P. and Wu, K.: Indexed Sequential Data Broadcasting in Wireless Mobile Computing, *Proc. 17th IEEE Int'l Conf. on Distributed Computing Systems*, pp.124–131 (1997).
9) Hsu, C., Lee, G. and Chen, A.: Index and Data Allocation on Multiple Broadcast Channels Considering Data Access Frequencies, *Proc. Third Int'l Conf. Mobile Data Management (MDM'02)* (2002).
10) Imielinski, T. and Badrinath, B.: Wireless Mobile Computing: Challenges in Data Management, *Comm. ACM*, Vol.37, No.10, pp.18–28 (1994).
11) Imielinski, T., Viswanathan, S. and Badrinath, B.: Power Efficient Filtering of Data on Air, *Proc. 4th International Conference on Extending Database Technology (EDBT)*, pp.245–258 (1994).
12) Jung, S., Lee, B. and Pramanik, S.: A Tree-Structured Index Allocation Method with Replication over Multiple Broadcast Channels in Wireless Environments, *IEEE Trans. on Knowledge and Data Engineering*, Vol.17, No.3, pp.311–325 (2005).
13) Lee, K., Leong, H. and Si, A.: Adaptive Semantic Data Broadcast in a Mobile Environment, *Proc. ACM Symposium on Applied Computing (SAC 2001)*, pp.393–400 (2001).
14) Leong, H. and Si, A.: Data Broadcasting Strategies over Multiple Unreliable Wireless Channels, *Proc. 4th Int'l Conf. on Information and Knowledge Management (CIKM'95)* (1995).
15) Lo, S. and Chen, A.: Optimal Index and Data Allocation in Multiple Broadcast Channels, *Proc. 16th Int'l Conf. Data Eng.*, pp.293–302 (2000).
16) Prabhakara, K., Hua, K. and Oh, J.: Multi-Level Multi-Channel Air Cache Design for Broadcasting in a Mobile Environment, *Proc. 16th Int'l Conf. Data Eng.* (2000).
17) Shivakumar, N. and Venkatasubramanian, S.: Efficient Indexing for Broadcast Based Wireless Systems, *Mobile Networks and Applications*, Vol.1, No.4, pp.433–446 (1996).
18) Imielinski, T., S.V. and Badrinath, B.: Data on Air: Organization and Access, *IEEE Trans. on Knowledge and Data Engineering*, Vol.9,

No.3, pp.353–372 (1997).

19) Tan, K. and Yu, J.: An Analysis of Selective Tuning Schemes for Nonuniform Broadcast, *Data and Knowledge Eng.*, Vol.22, No.3, pp.319–344 (1997).

20) Vaidya, N. and Hameed, S.: Scheduling Data Broadcast in Asymmetric Communication Environments, *ACM/Baltzer Wireless Networks*, Vol.5, No.3, pp.171–182 (1999).

21) Xu, J., Lee, W. and Tang, X.: Exponential Index: A Parameterized Distributed Indexing Scheme for Data on Air, *Proc. 2nd Int'l Conf. on Mobile Systems, Applications, and Services* (*ACM MobiSYS'04*), pp.153–164 (2004).

22) Yee, W. and Navathe, S.: Efficient Data Access to Multi-Channel Broadcast Programs, *Proc. ACM Conf. Information and Knowledge Management* (*CIKM'03*), pp.153–160 (2003).

23) Zipf, G.: *Human Behaviour and the Principle of Least Effort: An Introduction to Human Ecology*, Addison Wesley Press, Cambridge, Massachusetts (1949).

Damdinsuren Amarmend received his B.E. in computer science from Mongolian Technical University, Mongolia, in 1996, and his M.E. degree in computer science from Gunma University, Japan, in 2003. He is currently a Ph.D. student at the Department of Computer Science, Gunma University. His research interests include database systems, specially parallel and distributed database systems.

Masayoshi Aritsugi received his B.E. and D.E. degrees in computer science and communication engineering from Kyushu University, Japan, in 1991 and 1996, respectively. Since 1996, he has been working at the Faculty of Engineering, Gunma University, Japan, where he is now an Associate Professor. His research interests include database systems and parallel and distributed data processing. He is a member of IPSJ, IEICE, IEEE-CS, ACM, and DBSJ.

Yoshinari Kanamori received his D.E. degree from Tohoku University in 1975. Since 1991, he has been a Professor at the Department of Computer Science, Gunma University. His research interests include database systems and image processing. He is a member of IPSJ, IEICE, ACM, and IEEE-CS.