Distributed and Parallel Databases, 9, 151–177, 2001

A Hybrid Index Technique for Power Efficient Data Broadcast

QINGLONG HU* qinglong@cs.ust.hk
Department of Computer Science, University of Science and Technology, Clear Water Bay, Hong Kong

WANG-CHIEN LEE wlee@ieee.org
Verizon Laboratories Incorporated, 40 Sylvan Road, Waltham, MA 02451, USA

DIK LUN LEE dlee@cs.ust.hk
Department of Computer Science, University of Science and Technology, Clear Water Bay, Hong Kong

Received September 30, 1998; Revised March 18, 1999; Accepted July 1, 1999

Recommended by: Jin Jing

Abstract. The intention of power conservative indexing techniques for wireless data broadcast is to reduce mobile client tune-in time while maintaining an acceptable data access time. In this paper, we investigate indexing techniques based on *index trees* and *signatures* for data disseminated on a broadcast channel. Moreover, a *hybrid indexing* method combining strengths of the signature and the index tree techniques is proposed. Different from previous studies, our research takes into consideration of two important data organization factors, namely, *clustering* and *scheduling*. Cost models for the three indexing methods are derived for various data organization accommodating these two factors. Based on our analytical comparisons, the signature and the hybrid indexing techniques are the best choices for power conservative indexing of various data organization on wireless broadcast channels.

Keywords: wireless data broadcast, power conservation, index technique, signature method

1. Introduction

Due to resource limitations in a mobile environment, it is important to efficiently utilize wireless bandwidth and battery power in mobile applications. Wireless broadcasting is an attractive approach for data dissemination in a mobile environment since it tackles both bandwidth efficiency and power conservation problems [5, 13, 14, 18] On one hand, data disseminated through broadcast channels allows simultaneous access by an arbitrary number of mobile users and thus allows efficient usage of scarce bandwidth. On the other hand, the mobile computers consume less battery power when passively monitoring broadcast channels than actively interacting with the server by point-to-point communication.

Three criteria are used in this paper to evaluate the data access efficiency of broadcast channels:

*Present Address: IBM, Silicon Valley Lab, 555 Bailey Ave. San Jose, CA 95141, U.S.A.

- *Access Time*: The average time elapsed from the moment a client[1] issues a query to the moment when all the requested data frames are received by the client.
- *Tune-in Time*: The period of time spent by a mobile computer staying active in order to obtain the requested data.
- *Indexing Efficiency*: The tune-in time saved per unit of access time overhead for indexing.[2]

While access time measures the efficiency of access methods and data organization for broadcast channels, tune-in time is frequently used to estimate the power consumption by a mobile computer. Indexing efficiency, which correlates the access time and tune-in time, is used to evaluate the efficiency of indexing techniques in terms of minimizing the tune-in time while maintaining an acceptable access time overhead. In other words, a power conservative indexing technique has to balance out the index overhead (in terms of access time increased) and the tune-in time saved in order to maximize the indexing efficiency.

To facilitate efficient data delivery on broadcast channels, *scheduling* and *clustering* are frequently used to select and organize data for broadcast. Broadcast scheduling policies determine the content and organization of data broadcasting based on aggregate user data access patterns. *Broadcast disk* [1] is one of the well-known broadcast scheduling methods. In contrast, *flat broadcast* refers to the broadcast scheduling where each data frame is broadcast once in every cycle [9, 14]. When all frames with the same attribute value are broadcast consecutively, the data broadcast are called *clustered* on that attribute.[3] In contrast, the data broadcast are *non-clustered* on an attribute when all frames with the same value of that attribute are not broadcast consecutively. Clustering allows continuous reception of data with a specific attribute value.

Several indexing techniques for broadcast channels have been discussed in the literature [6, 13, 15, 17]. The basic idea behind these techniques is that, by including information about the arriving schedule of data frames in the broadcast channels, mobile computers are able to predict the arrival time of the requested data frames and thus, selective tuning in can be realized. Signature and index tree techniques [13, 15, 16] represent two different classes of indexing methods for broadcast channels. According to [13, 15], the index tree method is based on clustered data organization, while the signature methods don't presume a clustered data organization.[4] Moreover, indexing techniques used in these papers only took flat broadcast into consideration. In [13] index frames were treated in the same way as a data frame, although a better approach is to separate index frames from data frames. Lee and Lee [15] did not consider the clustered data. Although the authors demonstrated that the signature size played an important role in terms of data filtering efficiency and access latency, the optimal signature size was not given.

In this paper, we extend the existing works further. For the index tree method, since the size of an index frame is normally not equal to that of a data frame, to accurately estimate the access time and the tune-in time, we separate the index frames from data frames. For the signature method, we derive formulae to estimate the optimal signature size. In addition, scheduling and clustering are considered together with the index methods. The tune-in time and the access time cost formulae are developed to cover the cases: i) flat scheduling and clustered; ii) flat scheduling and non-clustered; and iii) broadcast disks scheduling and clustered. To our knowledge, there is no systematical study and comparison of the power conservative indexing techniques in the literature which takes both clustering and scheduling issues into account. In addition, we propose a new indexing method, called *hybrid index*, which takes strengths of both index tree and signature methods. Those three index methods are evaluated based on our criteria, namely, access time, tune-in time and indexing efficiency. Our results show that clustering and scheduling have major impacts on data organization of wireless data broadcast. We also conclude that the hybrid and signature methods give superior performance to the index tree method for various broadcast data organization accommodating the clustering and scheduling factors.

The rest of the paper is organized as follows. Section 2 gives an informal introduction of the broadcast channels, indexing techniques and system parameters used in performance evaluation and comparisons. In Section 3, indexing techniques based on index tree and signature methods are re-examined by taking the clustering and scheduling factors into consideration. In Section 4, a hybrid index scheme and corresponding cost models for access time and tune-in time are developed. Section 5 evaluates the indexing techniques in terms of tune-in time, access time, and indexing efficiency. Section 6 is a review of related work. Finally, Section 7 concludes the paper.

2. The data organization for wireless broadcast

In this section, we briefly introduce the concept of broadcast channels and some of the terminologies used. We assume that a base station is serving the role of an information server which maintains various kinds of multimedia data, including texts, images, audio/video and other system data. The server periodically broadcasts, on a specific channel, data of popular demands as a series of *data frames* to a large client population. These data frames vary in size and each frame consists of *packets* which are the physical units of broadcast. The header of a frame contains signals for synchronization and meta-information such as the type and the length of the frame. Logically the data frames are classified into two types: *record frames* and *index frames*, where record frames contain data items and index frames contain indexing information such as index tree nodes or signatures for a set of data items. Those two types of frames are interleaved together in a cycle. The clients retrieve the frames of their interest off the air by monitoring the broadcast channel. Since a set of data frames is periodically broadcast, a complete broadcast of the set of data frames is called a *broadcast cycle*. The organization of data frames in a broadcast cycle is called a *broadcast schedule*.

Data organization on the broadcast channels have great impacts on data access efficiency and power consumption. Data frames can be clustered based on attributes. Based on the results in [13], index tree techniques result in more efficient access for clustered information than non-clustered one. In this paper, we take both clustered and non-clustered data organization into consideration. Generally speaking, a non-clustered data organization can be divided into a number of segments, called *meta segments* [13], with non-decreasing or non-increasing values of a specific attribute. These meta segments can be considered as clustered and thus the indexing techniques for clustered data can be applied to them. To facilitate our study, we use the *scattering factor* M, the number of meta segments in the data organization, to model the non-clusterness of a data organization.[5]

Since access latency is directly proportional to the volume of data being broadcast, the volume of data in a broadcast cycle should be limited in such a way that only the most frequently accessed data frames are broadcast on the channel while the remaining data frames can be requested on demand through point-to-point connections [4, 9, 14, 18]. The server must determine the set of data frames to be broadcast by collecting statistics about data access patterns.

Due to some timely events, the client access pattern sometimes shows skewed distributions, which may be captured by *Zipf* or *Gaussian* distribution functions. In this case, scheduling data frames in broadcast disks (refer to Appendix and [1] for details) can achieve a better performance in terms of the access time and is a very important technique. As indicated in [1], in addition to performance benefits, constructing a broadcast schedule on multi-disks can give clients the estimated time when a particular data frame is to appear on the broadcast channel. This is particularly important for selected tune-in, data prefetching [3], hybrid pull and push technique [4, 18], and updates [2]. Therefore, the application of the index methods to broadcast disks is studied in this paper.

The broadcast disks method has better access time when the data frames with the same attribute values are clustered in one of the minor cycles. By receiving the cluster of data frames together, the mobile computer can answer the query without continuing to monitor the rest of broadcast cycle. This can be achieved by placing all of the data frames with the same indexed attribute value as a cluster on the same broadcast disk. The whole cluster of data frames are brought to the broadcast channel as a unit. Depending on speed of broadcast disks where this cluster is located, these data frames may appear several times in minor cycles. Thus, the resulting broadcast cycle is different from the completely clustered broadcast cycle. For broadcast scheduling adopting broadcast disks without using clustering, we simply consider the resulting broadcast cycle as non-clustered. In that case, broadcast disks lose their advantages over flat broadcast. Thus, when we consider index techniques for broadcast disks in the later sections of this paper, we only consider the clustered case. We assume only one broadcast channel since a channel with large bandwidth is logically the same as multiple channels with combined bandwidth of the same capacity. Moreover, it incurs smaller overheads of administrating than multiple channels. With the same token,we assume that index information is disseminated on the same broadcast channel. Finally, we assume that updates are only reflected between cycles. In other words, a broadcast schedule is fixed before a cycle begins.

Table 1 gives the parameters which describe the characteristics of a broadcast cycle. The cost models for the various index methods discussed later in this paper are derived based on these parameters. Although the sizes of data frames may vary, we assume frames

Table 1. System parameter setting.

D	Number of information frames (excluding index frames) in the broadcast cycle
F	Number of distinct information frames in the broadcast cycle ($F \leq D$)
P	Average number of packets in an information frame
S	Selectivity of query: average number of frames containing the same attribute value
M	Scattering factor of an attribute, which is the number of meta segments of the attribute

to be a multiple of the packet size. Both access time and tune-in time are measured in terms of number of packets. Before we develop the cost models for various index methods in broadcast disks, we derived a theorem[6] for the optimal broadcast scheduling based on multiple broadcast disks (please refer to Appendix for proof). The broadcast schedules derived from the theorem is used in our analysis later.

Theorem 1 (Optimal multi-disk broadcast)**.** *Given the number of data frames to be broadcast, D, the number of disks, N, the size of disk i, F_i where $1 \leq i \leq N$ and $\sum_{i=1}^{N} F_i = F$, and the broadcast frequency[7] of disk i, f_i where $1 \leq i < N$, $f_i > f_{i+1}$, broadcasting a data frame d on disk i, $\forall i \in [1, \ldots, N]$, in* fixed inter-arrival time *can achieve the optimal access time. In that case, to retrieve data frame d from disk i, client needs to scan, on an average of, $\frac{D}{2f_i}$ frames.*

3. Basic indexing techniques

In this section we discuss the basic ideas behind the index tree and the signature methods. We describe the distributed indexing and integrated signature techniques because they are the best methods of their class for single attribute indexing. The analytical cost models for the access time and the tune-in time for clustered and non-clustered data broadcast are presented. Moreover, the application of these index techniques to broadcast disks is also considered. Due to space limitations, we don't give all the derivations of these cost models. Interested readers can refer to [13] and [15] for more details.

3.1. The index tree techniques

As with a traditional disk-based environment, index-tree methods [13] have been applied to data broadcasts on wireless channels. Instead of storing the locations of disk records, the arrival time of the data frames is kept in the leaves of an index tree. The access method for retrieving data frames with an index tree technique involves the following steps:

- Initial probe: The client tunes into the broadcast channel and determines when the next index tree is broadcast.
- Search: The client follows a list of pointers to find out the arrival time of the desired data frames. The number of pointers retrieved is equal to the height of the index tree.
- Retrieve: The client tunes into the channel and downloads all the required data frames.

Figure 1 depicts an example of an index tree for a broadcast cycle which consists of 81 data frames [13]. The lowest level consists of square boxes which represent a collection of three data frames. The index tree is shown above the data frames. Each index node has three pointers.[8]

Table 2 gives the parameter setting for the index tree cost model. To reduce access time while maintaining a similar tune-in time for the client, the index tree can be replicated and interleaved with the information. *Distributed indexing* is actually one index replication and interleaving method. The index tree is broadcast every $\frac{1}{d}$ of the file during a broadcast

Table 2. Parameter setting for index tree schemes.

h	Height of the whole index tree
t	Number of upper levels in the index tree that are replicated
T	Number of packets in an index tree node
n	Number of search keys plus pointers that a node can hold

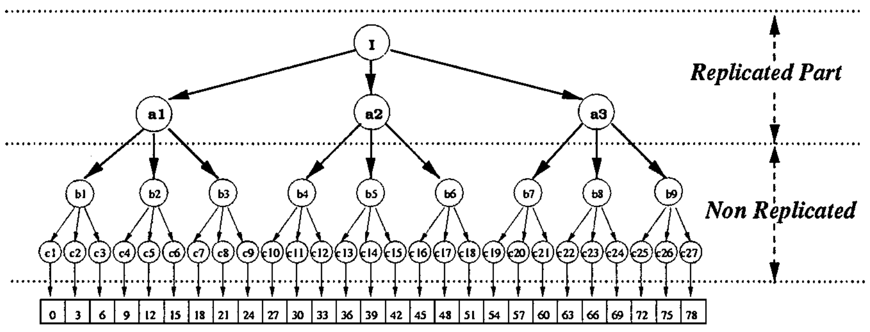

Figure 1. A full index tree.

cycle. However instead of interleaving the entire index tree d times, only the part of the index tree which indexes the data block immediately following it is broadcast. The whole index tree is divided into two parts: replicated and non-replicated parts. The replicated part constitutes the upper t levels of the index tree and each node in that part is replicated a number of times equal the number of children it has, while the non-replicated part consists of the lower $(h - t)$ levels and each node in this part appears only once in a given broadcast cycle. Since the lower levels of an index tree take up much more space than the upper part (i.e., the replicated part of the index tree), the index overheads can be greatly reduced if the lower levels of the index tree are not replicated. In this way, access time can be improved significantly without much deterioration in tune-in time.

To support distributed indexing, every frame has an *offset* to the root of the next index tree. The first node of each distributed index tree contains a tuple, with the first field as the primary key of the record that was broadcast last and the second field as the offset to the beginning of the next broadcast cycle. This is to guide the clients that have missed the required record in the current cycle to tune to the next broadcast cycle. There is a *control index* at the beginning of every replicated index to direct the client to a proper branch in the index tree. This additional index information for navigation together with the sparse index tree provides the same function as the complete index tree.

We assume that each node of the index tree takes up T packets and $X[h]$ and $X[t]$, respectively, are the total number of nodes of the full index tree and the replicated part of the index tree. The number of nodes in the i-th level of the index tree is denoted as $L[i]$.

In the distributed index tree, each node, p, in the replicated part is repeated as many times as the number of children that p has. Thus, the root is broadcast $L[2]$ times and nodes at level 2 are broadcast $L[3]$ times etc. For the index tree in Figure 1, since each node has three children, the root and nodes at level 2, i.e., a_1, a_2, and a_3, are broadcast 3 and 9 times, respectively. Therefore, in a broadcast cycle, the total number of nodes in the replicated part is $\sum_{i=2}^{t+1} L[i] = X[t+1]-1$. Additionally, the number of index nodes that are located below the t-th level of the index (i.e., the non-replicated) is, $X[h] - X[t]$. Hence, the total number of index nodes in a cycle is, $X[h]-X[t]+X[t+1]-1$ which equals to $X[h]+L[t+1]-1$. As a result, the index overhead is $T \cdot (X[h] + L[t+1] - 1)$ packets.

In the above discussion, we assumed that the file is clustered. For a non-clustered broadcast cycle, we can still apply index tree techniques to each meta segment. Instead of using one index tree for the entire broadcast cycle, an index tree is created for each meta segment. However, each index tree indexes all the values of the non-clustered attribute rather than indexing just the attribute values that appear in the current meta segment. For attribute values that do not appear in the current meta segment, a pointer in the index tree points to the next occurrence of the data frame with the desired attribute values. Thus, there are M distinct index trees for a broadcast cycle consists of M meta segments, The total overhead for putting index trees in a broadcast cycle is $T \cdot M \cdot (X[h] + L[t+1] - 1)$ packets.

To simplify the cost models, we average the index tree overhead to each data frame so that the size of a frame is considered to consist of a data part and an index overhead part. Of course, the actual index tree overhead for each data frame is different, but from the statistics point of view we can assume that all data frames have the same average index tree overhead. The average overhead for each data frame is $\mathit{TREE} = T \cdot M \cdot (X[h] + L[t+1] - 1)/D$. The replicated index tree part is broadcast every $1/L[t+1]$ fraction of each meta segment. Therefore, a broadcast cycle is divided into $M \cdot L[t+1]$ data blocks with replicated index nodes at the beginning of each block. Let P be the average number of packets for a data frame, the length of each block is $(\mathit{TREE} + P) \cdot D/(M \cdot L[t+1]) = ((X[h] + L[t+1] - 1) \cdot M \cdot T + D \cdot P)/(M \cdot L[t+1])$.

Flat Broadcast: Let us derive the access time and the tune-in time estimates for flat broadcast first. Since each frame is broadcast once in a cycle, the number of data frames in the broadcast, D, is equal to the number of distinct frames F. The initial probe period is the time to reach the index frame at the beginning of the next data block and can be estimated as:

$$\mathit{PROBE} = \frac{((X[h] + L[t+1] - 1) \cdot M \cdot T + F \cdot P}{2 \cdot M \cdot L[t+1]}$$

For a clustered broadcast, the scattering factor, $M = 1$ and the expected number of data frames before the arrival of the desired frames is $C = F/2$. Hence, the access time is:

$$\begin{aligned}\mathit{ACCESS} &= \text{initial probe time} + \text{waiting time before first desired frame arrives}\\ &\quad + \text{waiting time for retrieving all desired frames in the broadcast}\\ &= \mathit{PROBE} + \left(\frac{(X[h] + L[t+1] - 1) \cdot T}{F} + P\right) \cdot C + S \cdot P\end{aligned}$$

For a non-clustered broadcast cycle ($M > 1$), the access time is:

$$\begin{aligned}\mathit{ACCESS} &= \text{initial probe time}\\ &\quad + \text{waiting time for retrieving all desired frames in the broadcast}\\ &= \mathit{PROBE} + (X[h] + L[t+1] - 1) \cdot M \cdot T + F \cdot P\end{aligned}$$

The tune-in time for both clustered and non-clustered broadcast cycle depends on the initial probe, the scanning of index tree, the extra scanning of index tree in subsequent meta segments, and the retrieval of S data frames. Thus, the tune-in time of the index technique is upper bounded by:

$$TUNE = (h+1) \cdot T + (M+S) \cdot P$$

For a fully balanced indexing tree, the height of the tree, the number of nodes at i-th level, and the number of nodes in the upper t levels of the index tree are:

$$\begin{aligned} h &= \left\lceil log_n\left(\frac{F}{S}\right)\right\rceil \\ L[i] &= n^{i-1} \\ X[t] &= \sum_{i=0}^{t-1} n^i \end{aligned}$$

According to [13], the optimal height of the replicated part of the index tree for a broadcast, denoted as $\hat{t}$, can be estimated as:

$$\hat{t} = \left\lfloor \frac{1}{2} * \left(log_n\left(\frac{F}{S} + \frac{n^{h+1}}{n-1}\right) - 1\right)\right\rfloor + 1 \tag{1}$$

while for a non-clustered broadcast cycle, the optimal number of replicated levels $\hat{t}$ within a meta segment is:

$$\hat{t} = \left\lfloor \frac{1}{2} * \left(log_n\left(\frac{F*(n-1) + M \cdot n^{h+1}}{2 \cdot S \cdot M^2 \cdot (n-1)}\right) - 1\right)\right\rfloor + 1 \tag{2}$$

Broadcast Disks: For broadcast disks, as discussed in Section 2, data frames with the same attribute values are clustered inone minor cycle. In this case, we can treat each minor cycle of the broadcast disks as a meta segment.[9] An index tree can be built for each minor cycle. Similar to flat broadcast, the initial probe period, the time to reach the index frame at the beginning of the next data block, can be estimated as:

$$PROBE = \frac{((X[h] + L[t+1] - 1) \cdot M \cdot T + D \cdot P}{2 \cdot M \cdot L[t+1]}$$

where the number of data frames in the broadcast is $D = \sum_{i=1}^{N} F_i \cdot f_i$ and the scattering factor, M, is equal to the number of minor cycles in the broadcast (i.e., the LCM of the relative frequency of the disks). Hence, the access time for a clustered broadcast is:

$$\begin{aligned} ACCESS &= \text{initial probe time} + \text{waiting time before first desired frame arrives} \\ &\quad + \text{waiting time for retrieving all desired frames in the broadcast} \\ &= PROBE + \left(\frac{(X[h] + L[t+1] - 1) \cdot M \cdot T}{D} + P\right) \cdot C + S \cdot P \end{aligned}$$

Note that in the above equation, based on Theorem 1, the expected number of data frames before the arrival of the desired frames is $C = \frac{D}{2f_i}$ and the optimal number of replicated levels within a minor cycle can be derived from Eq. (2).

Since all the desired data frames are clustered in one minor cycle, the tune-in time is the same as in flat broadcast for a clustered broadcast cycle, i.e.,

$$TUNE = (h + 1) \cdot T + (1 + S) \cdot P.$$

3.2. The signature technique

Signature methods have been widely used for information retrieval. A signature of a data frame is basically a bit vector generated by first hashing the values in the data frame into bit strings and then superimposing them together. The signature technique interleaves signatures with their associated data frames in data broadcasting [15, 16].

To answer a query, a query signature is generated in a similar way as a data frame signature based on the query specified by the user. The client simply retrieves information signatures from the broadcast channel and then matches the signatures with the query signature by performing a bitwise *AND* operation. When the result is not the same as the query signature, the corresponding data frame can be ignored. Otherwise, there are two possible cases. First for every bit set in the query signature, the corresponding bit in the data frame signature is also set. This case is called *true match*. Second the data frame in fact does not match the search criteria. This case is called *false drop*. Obviously the data frames still need to be checked against the query to distinguish a true match from a false drop.

The primary issue with different signature methods is the size and the number of levels of the signatures. The access method for a signature scheme involves the following steps:

- Initial probe: The client tunes into the broadcast channel for the first received signature.
- Filtering: The client accesses the successive signatures and data frames to find the required data. On an average, it takes half of a broadcast cycle for the client to get the first frame with the required attribute.
- Retrieve: The client tunes in to get the successive desired data frames from the channel.

The number and the size of the signatures and the average false drop probability of the signatures[10] affect tune-in time and access time. The average false drop probability may be controlled by the size of the signatures. The initial probe time is related to the number of signatures interleaved with the data frames. Table 3 defines the parameters for signature cost

Table 3. Parameter setting for signature scheme.

P_f	false drop probability for integrated signatures
k	number of information frames indexed by an integrated signature
p	number of bits in a packet
R	the size (number of packets) of an integrated signature

models. Estimation of the average false drop probability is given in the following Lemma [15]:

Lemma 1 (optimal false drop probability). *Given the size of a signature, R, the number of bit strings superimposed into the signature, s, the average false drop probability for the signature is, $P_f = 2^{-R\cdot(p\cdot ln2)/s}$.*

In [15, 16], three signature algorithms, namely *simple signature*, *integrated signature*, and *multi-level signature*, were proposed and their cost models for access time and tune-in time were given. For simple signatures, the signature frame is broadcast before the corresponding data frame. Therefore, the number of signatures is equal to the number of data frames in a cycle. An integrated signature is constructed for a group of consecutive frames, called a *frame group*. The multi-level signature is a combination of the simple signature and the integrated signature methods, in which the upper level signatures are integrated signatures and the lowest level signatures are simple signatures. Since the three signature algorithms have been extensively compared in the literature [8, 15, 16], we don't repeat the comparisons here. In the context of this study, simple signature is not very efficient since it will be generated from only one attribute. Thus, we select the integrated signature method to compare with the index tree method and the new index methods proposed later in this paper.

Figure 2 illustrates an integrated signature scheme. An integrated signature indexes all of the data frames between itself and the next integrated signature. The integrated signature method is general enough to accommodate both clustered and non-clustered data broadcast. For clustered data broadcast, a lot of data frames can be indexed by one integrated signature. According to Lemma 1, the smaller the number of bit strings s superimposed into an integrated signature, the lower the false drop probability. The integrated signature generated for a clustered broadcast cycle has the effect of reducing the number of bit strings superimposed. To maintain a similar false drop probability for a non-clustered broadcast cycle, the number of data frames indexed by an integrated signature may be reduced. Determining the number of data frames for signature generation requires further study.

To simplify our discussion, we assume that frames with the same attribute value for an attribute a are evenly distributed in each meta segment. Consequently, the number of frames with the same attribute value in each meta segment is $\lceil S/M \rceil$, where the attribute a has a selectivity S and a scattering factor M. Let k be the number of data frames indexed by an integrated signature. The number of distinct attribute values used for signature generation,

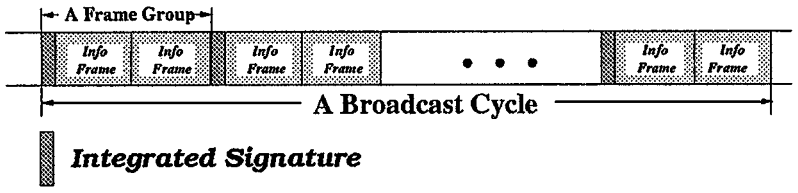

Figure 2. An example of the integrated signature technique.

s, can be estimated as $\lceil k/\lceil S/M \rceil \rceil$. For frames in a meta segment, the average number of qualified frames corresponding to a matched integrated signature, called locality of true matches l ($1 \le l \le k$), can be estimated as, $l = k/\lceil k/\lceil S/M \rceil \rceil$, for frames which are randomly distributed over the file, l is equal to 1.

Flat Broadcast: Next, we derive the access time and the tune-in time for clustered and non-clustered broadcast cycles. Let *SIG* be the average signature overhead for each data frame. Then, $SIG = R/k$. Once again, we assume that the expected number of data frames before the arrival of the desired frames is C.

For clustered data broadcast, the access time can be derived as follows:

$$\begin{aligned} ACCESS &= \text{initial probe time} + \text{waiting time before first desired frame arrives} \\ &\quad + \text{waiting time for retrieving all the desired frames in the broadcast} \\ &= (R + k \cdot P)/2 + (SIG + P) \cdot C + S \cdot P \end{aligned} \tag{3}$$

and the tune-in time is:

$$\begin{aligned} TUNE &= \text{tune-in time in initial probe period} + \text{ true match frames in the broadcast} \\ &\quad + \text{integrated signatures before the first desired frame} \\ &\quad + \text{false drop frames before the first desired frame} \\ &= P/2 + P \cdot (S + k/2) + (R/k + P_f \cdot P) \cdot C \end{aligned} \tag{4}$$

For a non-clustered broadcast cycle, the access time is:

$$\begin{aligned} ACCESS &= \text{initial probe time} + \text{waiting time for the first desired frame to arrive} \\ &\quad + \text{waiting time for retrieving all the desired frames in the broadcast} \\ &= (R + k \cdot P)/2 + SIG \cdot F + P \cdot F \end{aligned}$$

and the tune-in time is:

$$\begin{aligned} TUNE &= \text{tune-in time in initial probe period} + \text{true match frames in the broadcast} \\ &\quad + \text{integrated signatures for the retrieval of all the desired frames} \\ &\quad + \text{false drop frames for the retrieval of all the desired frame} \\ &= P/2 + F \cdot R/k + (k/l \cdot k/2 + S + P_f \cdot F) \cdot P. \end{aligned} \tag{5}$$

According to Lemma 1, we have, $P_f = 2^{-(p \cdot ln2/s) \cdot R}$. We differentiate Eq. (4) or (5) with respect to R and let $\partial TUNE/\partial R$ equal zero. Then the optimal signature size (number of packets), $\hat{R}$, can be computed as:

$$\hat{R} = \frac{s}{p \cdot ln2} \cdot log_2 \frac{k \cdot P \cdot p \cdot ln^2 2}{s} \tag{6}$$

Broadcast Disks: For broadcast disks, the access time and the tune-in time can also be obtained by Eqs. (3) and (4) respectively. Compared with flat broadcast, the difference is in the parameter C, i.e., for flat broadcast $C = F/2$, for broadcast disks $C = \frac{D}{2f_i}$ and $D = \sum_{i=1}^{N} F_i \cdot f_i$.

4. The hybrid index approach

Both the signature and the index tree techniques have advantages and disadvantages in one aspect or the other. For example, the index tree method is good for random data access, while the signature method is good for sequentially structured media such as broadcast channels. The index tree technique is very efficient for a clustered broadcast cycle, but the signature method is not affected much by clustering factor. While the signature method is particularly good for multi-attribute retrieval, the index tree provides a more accurate and complete global view of the data frames based on its indexed value. Since the clients can quickly search in the index tree to find out the arrival time of the desired data, the tune-in time is normally very short. Since a signature does not contain global information about the data frames, it can only help the clients to make a quick decision on whether the current frame (or a group of frames) is relevant to the query or not. The filtering efficiency heavily depends on the false drop probability of the signature. As a result, the tune-in time is normally high and is proportional to the length of the broadcast cycle.

In this section, we develop a new index method, called *hybrid index*, which builds on top of signatures a *sparse* index tree to provided global view for data frames and their corresponding signatures. A *key-search pointer* node in the sparse index tree points to a *data block* of consecutive frames and their corresponding signatures (refer to figure 3).

The index tree is called sparse tree because only upper t levels of the whole index tree are constructed. Obviously, the sparse index tree overhead depends on t. The larger the t, the more precise location information the sparse tree provides, and the higher the access time overhead. One extreme case is t equals h the number of the whole index tree levels. The hybrid index evolves to the index tree method. On the other hand, if t equals zero, the hybrid index method becomes the signature method.

To retrieve information, the client can search the sparse index tree to obtain the approximate location information about the desired data frames. Since the size of the upper

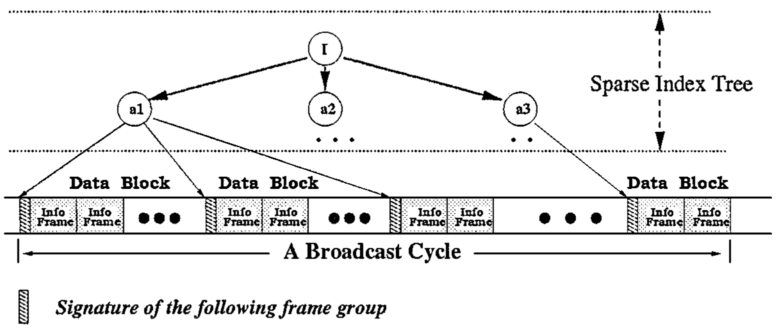

Figure 3. The hybrid of index tree and signature.

t levels of an index tree is usually small the overhead for this additional index is very small.

Since the hybrid index technique is built on top of signature method, it retains all of the advantages that a signature method has. However, the global information provided by the sparse index tree improves tune-in time considerably. The general access method for retrieving data with this technique now becomes:

- Initial probe: The client tunes into the broadcast channel and determines when the next index tree arrives.
- Search: Upon receipt of the index tree, the client accesses a list of pointers in the index tree to find out when to tune into the broadcast channel to get to the nearest location where the required data frames can be found.
- Filtering: At the nearest location, a successive signature filtering is carried out until the desired data frames are found.
- Retrieval: The client tunes into the channel and downloads all the required data frames.

4.1. Cost model analysis

Based on the above definition of the hybrid indexing method, we derive an estimates of the access time and the tune-in time. The sparse index tree is the same as the replicated part of the index tree method. The average waiting time for retrieving one data frame from the broadcast cycle with M meta segments can be expressed as:

$$TREE + SIG + P = T \cdot M \cdot (X[t+1] - 1)/D + R/k + P$$

where $TREE$ and SIG are the index overheads of the index tree parts and the signature parts of a frame. The average number of data frames in one data block $D[B]$ can be calculated in a similar way as in the index tree method, which is $D/(M \cdot L[t+1])$. Thus, the total index tree and signature overheads in a data block are $D[B] \cdot TREE$ and $D[B] \cdot SIG$, respectively. Hence, the average initial probe time for the index tree is half of the data block:

$$PROBE = (TREE + SIG + P) \cdot D[B]/2$$

Flat Broadcast: For the clustered broadcast cycle with flat broadcast scheduling, the expected access time for hybrid indexing method is:

$$ACCESS = PROBE + (TREE + SIG + P) \cdot C + S \cdot P \tag{7}$$

where $D = F$ and the expected number of data frames before the arrival of the desired frames C is $F/2$. If the broadcast cycle is non-clustered, then there is one sparse index tree for each meta segment. Index tree technique is applied in each meta segment. Hence, the expected access time is:

$$ACCESS = PROBE + (TREE + SIG + P) \cdot F$$

For both clustered and non-clustered broadcast cycle with flat broadcast scheduling, the tune-in time primarily depends on the initial probe of the client to determine the next occurrence of the control index, the access time for the index tree part which equals to the number of levels t of the sparse index tree, the tune-in time for the data block B, the selectivity of a query S, and the successive access to M meta segments. Therefore, it is upper bounded by:

$$TUNE = (t+1) \cdot T + (M+S) \cdot P + TUNE_B \tag{8}$$

where $TUNE_B$ is defined as the tune-in time for filtering data block B with the signature technique. It can be estimated as follows:

$$\begin{aligned} TUNE_B = {} & \text{every signature in half the length of the data block B} \\ & + \text{ false-drop data frames in half the length of the data block B} \end{aligned}$$

Broadcast Disks: For broadcast disks, the access time for hybrid indexing method can be obtained by Eq. (7) (i.e., for broadcast disks $C = \frac{D}{2f_i}$ and $D = \sum_{i=1}^{N} F_i \cdot f_i$). The tune-in time of broadcast disks is the same as that of flat broadcast for a clustered cycle (i.e., let $M = 1$ in Eq. (8)).

Note: According to Eq. (8), the tune-in time is proportional to M. Hence, the hybrid method is efficient only for the broadcast cycle with small M. Actually, the sparse index tree introduces overhead for the non-clustered broadcast cycle with large M. In this case, retrieval based on signatures can result in better tune-in time. However, the hybrid method supports multi-attribute indexing very well [10]. For an attribute with small scattering factor, a sparse index tree can be built to reduce the tune-in time. For an attribute with high scattering factor, there is no need to build the sparse index tree and the client simply filters out the requested data frames sequentially and ignores the sparse index tree. We extend the hybrid index with control information, which includes the size of the sparse index tree and the size of the data block. When a query is specified on a non-clustered attribute, this control information is used to direct the client to the beginning of the next data block. Starting there, the client matches the signatures one by one for each data frame in that data block. Hence, the access time for non-clustered information is the same as Eq. (7). In order to skip each of these index trees, we assume that the client needs to retrieve an index node to get information such as the size of the sparse index tree and the size of the data block. Therefore, the tune-in time is:

$$TUNE = P + TUNE_{Sig}$$

where $TUNE_{Sig}$ is defined to be the tune-in time for the corresponding signature scheme used (i.e., integrated signature in this paper).

5. Evaluation of index methods

In this section, we compare the access time, the tune-in time, and the indexing efficiency of the index tree, the integrated signature, and the hybrid techniques. We also include the case where no index is used (denoted as *non-index*) as a baseline for comparisons. Our comparisons are based on the cost models developed previously. Orthogonal to the index method, frames can be broadcast based on broadcast disks or flat broadcast. Thus, there are various combinations to be considered.

For flat broadcast, each data frame appears once in a given broadcast cycle. Therefore, the number of data frames in the broadcast D equals the number of distinct frames F. For a clustered broadcast cycle (i.e., $M = 1$), on average, half of a broadcast cycle needs to be scanned before the desired frames arrive (i.e., $C = F/2$).

For broadcast disks (M = number of minor cycles), D is greater than F due to frame duplication in the broadcast cycle. The access time and the tune-in time on different disks i may be different. We denote the average access probability, the access time, and the tune-in time for frames on disks i as P_i, $Access_i$, and $TUNE_i$, respectively. For disk i with frequency f_i, the expected number of frames scanned before the arrival of the desired frames, C, is given by Theorem 1. Therefore, the estimates for the average access time and tune-in time are:

$$ACCESS = \sum_{i=1}^{N} ACCESS_i \cdot P_i$$

$$TUNE = \sum_{i=1}^{N} TUNE_i \cdot P_i$$

The study for a non-clustered broadcast cycle is especially important in multi-attribute indexing where cycle can be clustered on at most one attribute, while query requests on other attributes get a reply via indexes built on the non-clustered broadcast cycle. For a non-clustered broadcast cycle, M is greater than 1 and the client needs to scan the entire broadcast to retrieve all the desired frames.

Table 4 lists the parameter values used in the comparisons. Both access time and tune-in time are measured in number of packets and are compared with respect to the number of distinct frames in a broadcast cycle which is varied from 10^3 to 10^6. We made the following assumption in the comparisons: a frame has capacity $P = 1000$ packets and a tree node takes up $T = 100$ packets which can contain $n = 10$ search keys and pointers, the size of a packet is $p = 128$ bits, four frames (i.e., $k = 4$) are grouped together in an integrated signature, the index tree is balanced (all leaves are on the same level) and each node has the same number of children. In order to make comparison, the sparse tree levels t of the hybrid method is set to the same as the replicated tree levels in the index tree method, which can be obtained via Eq. (1).

Table 4. Parameters of the cost models.

$F = 10^3$ to 10^6	$P = 1000$	$p = 128$	$n = 10$
$T = 100$	$k = 4$	$S = 1$	$M = 1$ to 200
$N = 3$	$F_{1,2,3} = F/10, 2F/5, F/2$	$f_{1,2,3} = 3, 2, 1$	$P_{1,2,3} = 1/3$

A broadcast cycle with selectivity $S > 1$ is logically equal to a broadcast cycle with selectivity $S = 1$ and the data frame size S times of the original broadcast cycle. Thus, in this paper, we only explore the case where the query selectivity S is 1.

For broadcast disks, we assume that three disks are adopted (i.e., $N = 3$). The sizes of fast, medium, and slow disk are, respectively, 1/10, 1/2.5, and 1/2 of the total number of frames and the relative spin speeds are 3, 2, and 1. The aggregate client access probability for each disk is the same (i.e., $P_i = 1/3$, $1 \leq i \leq N$). Within each disk, all data frames have equal average access probability. Therefore, the average access probability for each data frame is inversely proportional to the size of the disk where the data frame is located. For a non-clustered broadcast cycle, we vary the scattering factor M (i.e., from 1 to 200) to examine its impact on the performance of the index methods.

In what follows, we will first evaluate the access time, the tune-in time, and the indexing efficiency of index methods for clustered broadcast cycle and then for non-clustered broadcast cycle. For the clustered broadcast cycle, we consider both of the broadcast disks and flat broadcast as broadcast scheduling policies while for the non-clustered broadcast cycle, we only consider the flat broadcast scheduling.

5.1. *The clustered broadcast cycle*

In this section we study the access time and the tune-in time of the index methods for a clustered broadcast cycle. Figures 4 and 5 depict the access time and the tune-in time comparisons, where the y coordinate is in logarithmic scale and the access time is the overhead with respect to non-index for broadcast disks scheduling.

First we consider the access time in figure 4. The curves representing the access time overhead of the hybrid, the signature, and the non-index methods (denoted as *hybrid*, *sig*,

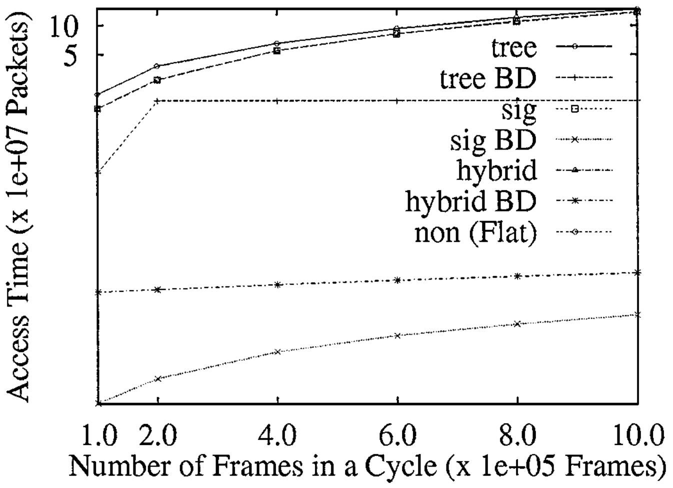

Figure 4. Access time overhead comparisons for clustered cycle.

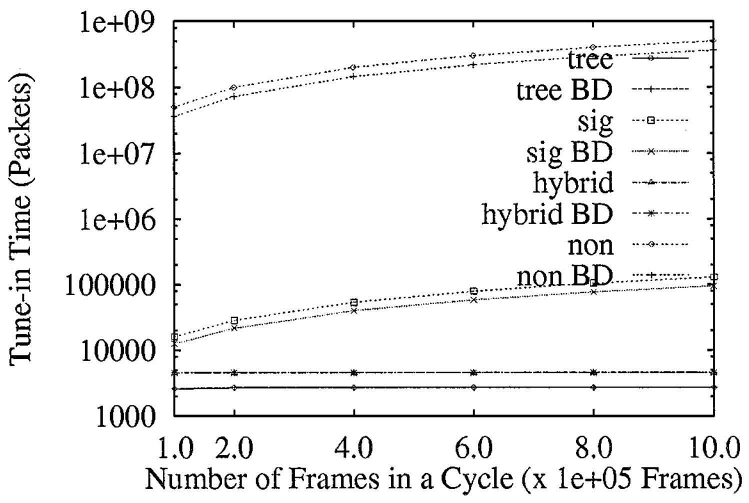

Figure 5. Tune-in time comparisons for clustered cycle.

and *non*, respectively) overlap each other for flat broadcast. Generally, amongst all broadcast scheduling and indexing methods, the non-index method with broadcast disks gives the shortest access time which is proportional to the size of a broadcast cycle. For any particular indexing methods, the access time for broadcast disks (denoted with BD in the figures) is always better than that for flat broadcast because of the skewed client access pattern.

When we consider flat broadcast only, the access time for the signature and the hybrid methods is similar to the non-index method as indicated by the overlapping curves in figure 4 while the access time for the index tree method gives an obviously worse access time. Compared with the non-index method, the index overhead for the index methods (especially the signature and the hybrid methods) does not deteriorate the access time much for a clustered broadcast cycle.

In the broadcast disks method, the broadcast cycle is longer than that scheduled in flat broadcast. Since the longer the broadcast cycle, the higher the index overhead, all three index methods give a much worse access time than the non-index BD. The signature method performs better than the hybrid and the tree methods. Since the index tree is replicated in every minor cycle, its index overhead for broadcast disks is the highest. Thus, the difference between the index tree method and the other two index methods for broadcast disks is much larger than that for flat broadcast.

Next, we consider the tune-in time of the index methods. Figure 5 shows that the curves representing the index tree method (denoted as *tree*) and the hybrid method are overlapping for both broadcast disks and flat broadcast. The non-index methods give much worse results than the index methods. This suggests that indexing can improve client tune-in time considerably. If we focus on the index methods only, the index tree method gives the best tune-in time and the signature method has the worst tune-in time. Broadcast disks can also improve the tune-in time of the index methods. As shown in figure 5, the broadcast disks

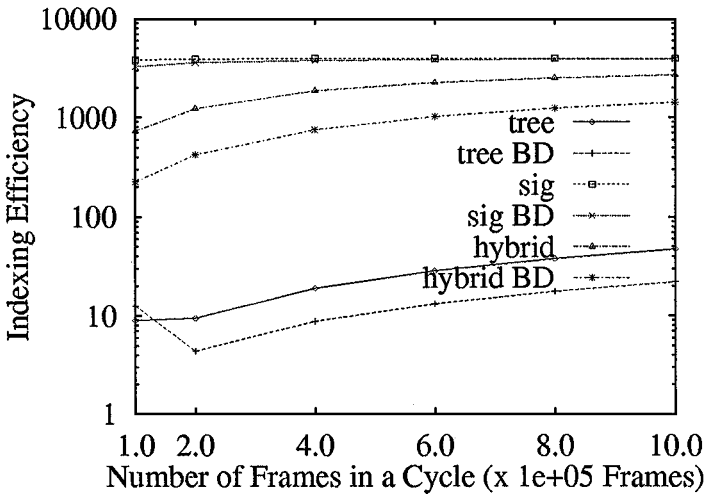

Figure 6. Indexing efficiency for clustered cycle.

improve the tune-in time of the index methods and such improvement for the non-index and the signature methods is more than for others.

In order to investigate the relationship between the tune-in time and the access time, we demonstrate in figure 6 the indexing efficiency of indexing methods for various sizes of a broadcast cycle. The tune-in time saved and the access time overhead is calculated with respect to the non-index method for broadcast disks and flat broadcast. Intuitively, the larger the amount of tune-in time saved per unit access time overhead, the better the index methods. We can observe that the amount of tune-in time saved per unit of access time overhead increases as the number of frames in a cycle increases. The figure tells us that the signature method can give the largest amount of tune-in time saved per unit of access time overhead and the index tree method gives the least amount of saving which is much less than the other two methods. Indexing broadcast disks results in less amount of tune-in time saved than the indexing flat broadcast.

In conclusion, when a broadcast cycle is clustered by attributes, the hybrid scheme is the best when the access time, the tune-in time, and indexing efficiency are considered. If only the tune-in time is considered, then the index tree scheme shows the best performance. If we consider the indexing efficiency, then the signature is the most efficient index method. Broadcast disks approach can improve both the access time and the tune-in time when the client access patterns are skewed, although the improvement in the tune-in time is not as significant as that in the access time.

5.2. *The non-clustered broadcast cycle*

In this section, we investigate the index methods for a non-clustered broadcast cycle (i.e., $M > 1$). To examine the influence of M on the system performance, we fix the number of

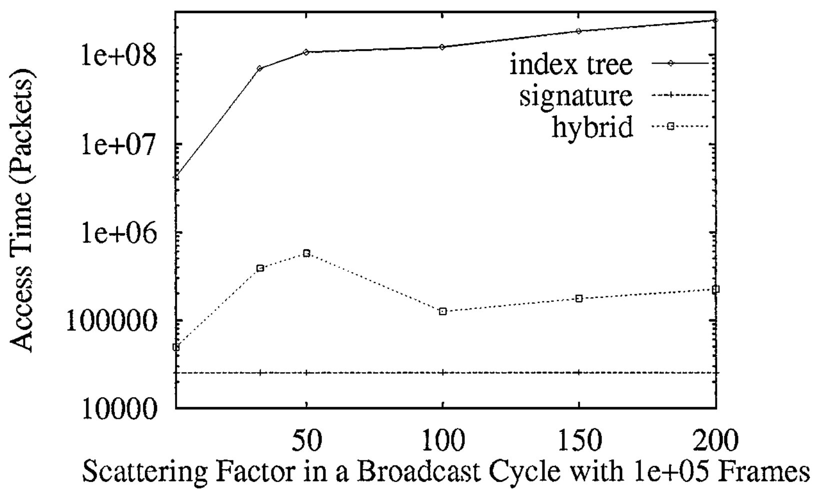

Figure 7. Access time overhead vs. scattering factor.

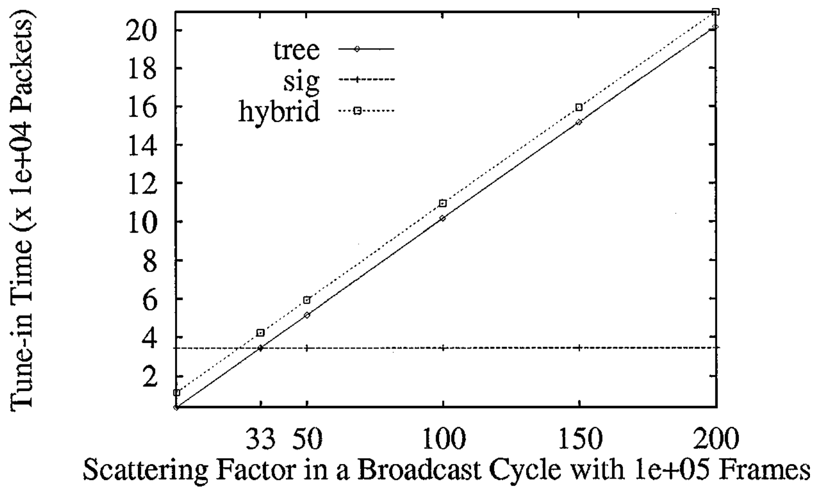

Figure 8. Tune-in time vs. scattering factor.

frames in a cycle to 10^5 and vary M from 1 to 200. The access time overhead is obtained with respect to the non-index method. Figures 7 and 8 illustrate the results. As expected, the scattering factor has great impact on the access time of the index tree method. Since there is an index tree corresponding to every meta segment, as M is increased, the index tree overhead increases rapidly. For the hybrid method, although there is a sparse index tree for each meta segment, the sparse index tree overhead is very small and as M increases, the initial probe time for index tree node decreases. Therefore, M has little influence on the access time in the hybrid method. As shown in figure 8, the tune-in time of the index tree

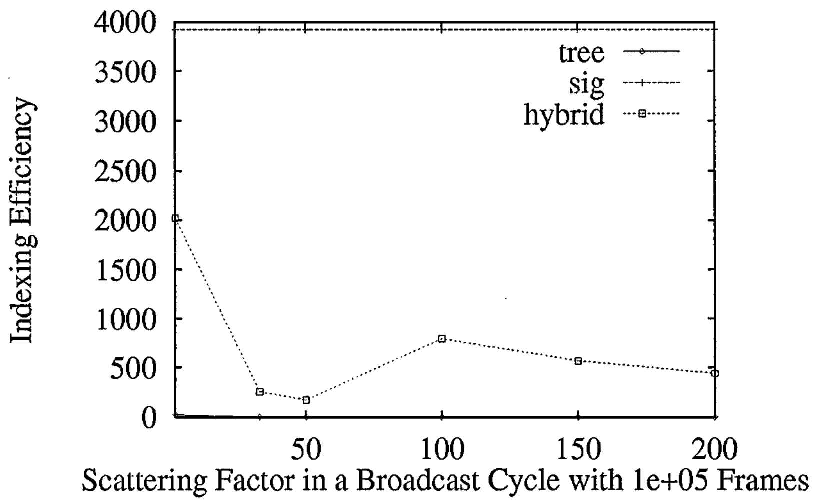

Figure 9. Indexing efficiency vs. scattering factor.

and the hybrid methods goes up quickly as M is increased, while the tune-in time of the signature index method remains the same. Since both the index tree and the hybrid methods need to probe each meta segment for the possible arrival of the desired frames, the major advantage of the index tree and the hybrid methods, namely, short tune-in time, disappears when M is greater than 33. However, there is no impact on the signature method for both the access time and tune-in time when the scattering factor changes. This suggests that the index tree and the hybrid methods are not applicable to a broadcast cycle with a large scattering factor.

Similar to the previous section, figure 9 depicts the indexing efficiency with respect to different scattering factors in a broadcast cycle. The tune-in time saved for the index tree is very low while the tune-in time saved for the signature method is the highest.

Finally, we use the same parameter settings as in the clustered broadcast cycle case, but we assume that the broadcast cycle is non-clustered with a scattering factor set to 100 (refer to figures 10 and 11). The access time overhead is obtained with respect to that of the non-index. Similar to the clustered broadcast cycle, the access time of the index tree method is much worse than that of the other two index methods. The signature method has the closest access time to the non-index method. Since we assume that M is fixed at 100 for any broadcast length and there is an index overhead for each meta segment, unlike the clustered cycle, the tune-in time of the index tree and hybrid methods is not always better than that of the signature method. That is, for small broadcast cycle (i.e., less than 4×10^5), the signature methods have the best tune-in time among the three methods. When the length of a cycle increases, the tune-in time of the signature method increases quickly due to false drops and becomes worse than the other methods again. As in the case of clustered cycle, the tune-in time of the hybrid method is always a little bit worse than that of the index tree method. In figure 12, we illustrate the indexing efficiency for various cycle lengths. All index methods display similar interrelation to that in the case of clustered cycle.

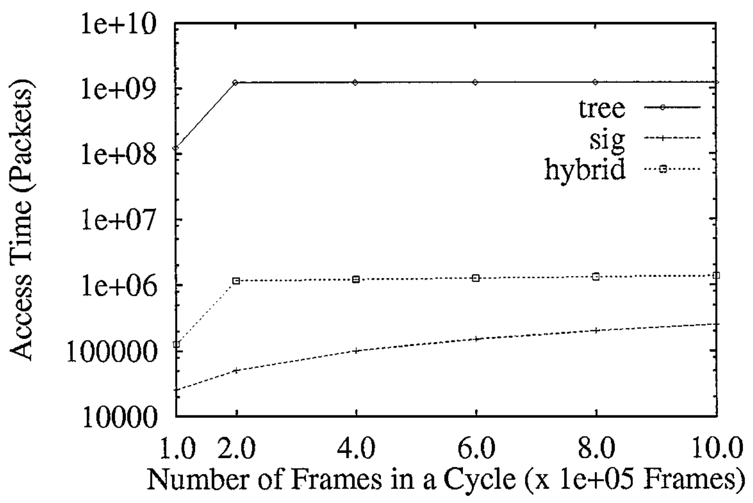

Figure 10. Access time comparisons for non-clustered cycle.

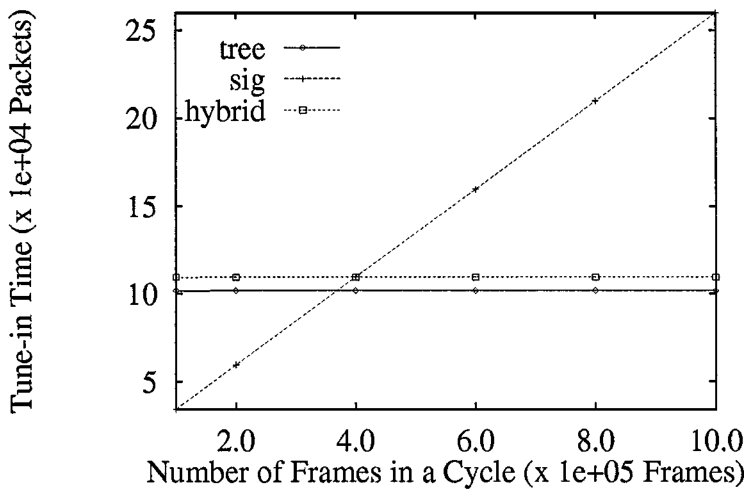

Figure 11. Tune-in time comparisons for non-clustered cycle.

For both clustered and non-clustered broadcast cycle, we observe that the tune-in time of the signature schemes is proportional to the length of the broadcast cycle, while the other two methods have the tune-in time independent of the length of the broadcast cycle. The reason is that the size of the index tree can be adjusted automatically according to the length of the broadcast cycle F and the height of the index tree h increases very slowly ($n^h \approx F$) and only h affects the tune-in time of the clients.

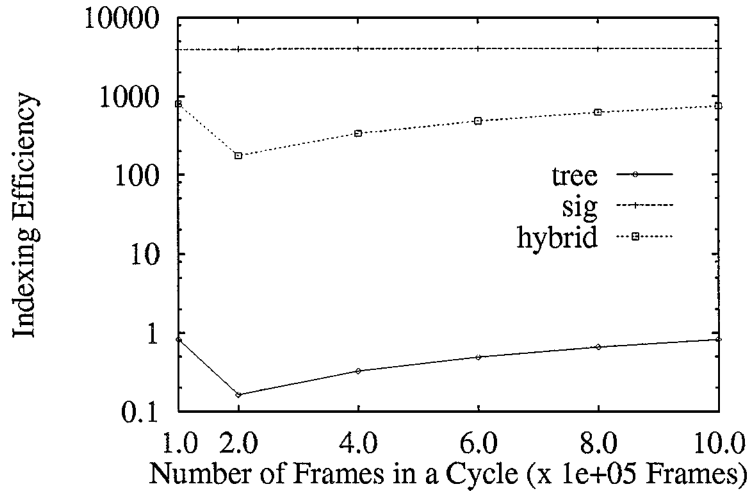

Figure 12. Indexing efficiency for non-clustered cycle.

6. Related works

The basic idea of constructing index on broadcast data was investigated by a number of projects [12, 13, 15]. To reduce the power consumption of clients, Imielinksi et al. [11, 13] proposed two methods, $(1, d)$ indexing and distributed indexing. In $(1, d)$ indexing method, the index tree is broadcast d times during one broadcast cycle. The full index tree is broadcast following every $\frac{1}{d}$ fraction of the file. All frames have an offset to the beginning of the next index segment. The first frames of each index segment has a tuple, with the first field as the attribute value of the record that was broadcast last and the second field as the offset to the beginning of the next cycle. This is to guide the clients that have missed the required frames in the current cycle and have to tune to the next cycle. Notice that there is no need to replicate the entire index between successive data segments, the distributed indexing techniques was developed, it interleaves and replicates index tree with data, in the sence that most frequently access index part (the upper level of the index tree) is replicated the number of times equal to the number of children.

The project in [12] discussed the hashing schemes and a flexible indexing method for organizing broadcast cycle. In the hashing schemes, instead of broadcasting a separate directory with the information frames, the hashing parameters are included in the frames. Each frame has two parts: the *data* part and the *control* part. The control part is the "investment" which helps guide searches to minimize the access time and the tune-in time. It consists of a hash function and a shift function. The shift function is necessary since most often the hash function is not perfect. In such a case there can be collisions and the colliding frames are stored immediately following the frame assigned to them by the hashing function. The flexible indexing method first sorts the data in ascending (or descending) order and then divides the cycle into p segments numbered 1 through p. The first frame in each of the data segments contains a control part consisting of the control index. The control index is a binary index which, for a given key K, helps to locate the frame which contains that key. In this way, we can reduce the tune-in time. The parameter p makes the indexing method flexible since depending on its value we can either get a very good tune-in time or a very good access time.

Lee and Lee [15] investigated the signature techniques for flat data broadcasting. Three signature methods, simple signature, integrated signature, and multi-level signature, were proposed and their cost models for the access time and the tune-in time were given. Based on the models, they made comparisons for the performance of different signature methods. Work in [16] explored the influencies of caching signatures in the client side to the system performance. Four caching strategies were developed and the tune-in time and the access time were compared. With reasonable access time delay, all the caching strategies help in reducing the tune-in time for the two-level signature scheme.

All those index methods can reduce the power consumption to some extent with a certain amount of access overhead. However, the index techniques developed previously didn't consider the characteristics of skewed access patterns.

Recent work in [6] developed an imbalanced index tree on broadcast data. The index tree is constructed in accordance with data access frequencies in such a way that the expected cost of index probes for data access is minimized. In contrast to [13], the variant fanouts for index nodes was also exploited. Since the cost of index probes takes up small part of the overall cost, such imbalanced index tree gives only limited improvement.

To reduce the overall access time, as mentioned in the introduction section, Broadcast disks [1, 20] is an efficient technique which can improve the overall access time for skewed data access patterns. In their later work, Acharya et al. [3] studied the opportunistic prefetching from broadcast disks by the client, Acharya et al. [2] considered the case when update presents in broadcast disks, and [4] studied the performance of a hybrid data delivery in broadcast disks environments, where clients can retrieve their desired data items either by monitoring broadcast channel (push-based) or by issuing explicit pull request to server (pull-based). These studies indicate that data prefetching and hybrid data delivery with caching can significantly improve performance over pure pull-based caching and pure push-based caching. While updates have no great influence on the system performance. Hameed and Vaidya [7] further developed an $O(log(n))$ time-complexity scheduling algorithm which can determines the broadcast frequency of each data item according to data access patterns for both single and multiple broadcast channels. In their models, the length of data items is not necessarily of the same. However, no study explores index on broadcast disks.

7. Conclusion and future work

In a mobile environment, power conservation of the mobile clients is a critical issue to be addressed. An efficient power conservative indexing method should introduce low access time overhead, consume low tune-in time, and produce high indexing efficiency. Moreover, an ideal index method should perform well under both clustered and non-clustered broadcast cycle, with different broadcast scheduling policies, such as flat broadcast and broadcast disks.

In this paper, we evaluate the performance of power conservative indexing methods based on index tree and signature techniques. Combining strengths of the signature and the index tree techniques, a hybrid indexing method is developed in this paper. This method has the advantages of both the index tree method and the signature method and has a better performance than the index tree method. A variant of the hybrid indexing method has been demonstrated to be the best choice for multiple attributes indexing organization in wireless broadcast environments [10].

Our evaluation of the indexing methods takes into consideration the clustering and scheduling factors which may be employed in wireless data broadcast. Access time, tune-in time, and indexing efficiency are the evaluation criteria for our comparisons. We develop cost models for access time and tune-in time of the three indexing methods and produce numerical comparisons under various broadcast organization based on the formulae.

Through our comparisons for both clustered and non-clustered data broadcast cycles, we find that the index tree method has low tune-in time only for the clustered broadcast cycles or the non-clustered broadcast cycles with low scattering factor. The index tree always produces high access time overhead. For a broadcast cycle with high scattering factor, the signature method is the best choice. Since the signature method needs further filtering to determine whether a data item really satisfies a query, the tune-in time for signature methods may be high. However, the variations of data organization for the broadcast channels have very limited impact on performance of the signature method. Moreover, the access time overhead is low. The hybrid method has the advantages of both the index tree method and the signature method. It performs well for clustered broadcast cycles or non-clustered cycles with low scattering factor (i.e., low tune-in time similar to the index tree method and low access time overhead similar to the signature method). If we only consider the indexing efficiency, the signature method has the best performance for various broadcast organization.

Finally, through our comparisons for flat broadcast and broadcast disks, we observe that broadcast disks can reduce the access time for any index methods and the tune-in time for the signature and non-index methods.

As a related study [10], we have studied the performance of multi-attribute index methods for wireless broadcast channels. Since the access time and the tune-in time of the index methods may be different for queries based on different attributes, We have estimated the average access time and tune-in time of the client according to the queries arrival rate for each attribute.

In the future, we plan to incorporate the index schemes with data caching algorithms to achieve an improved system performance and obtain a better understanding of the wireless broadcast systems.

Appendix

Broadcast disks were proposed to improve data access efficiency [1]. The idea is to divide data frames to be broadcast into broadcast disks based on their access frequency and then interleave data frames on these disks into an information stream for broadcast. This imitates multiple disks each spinning at a different speed. The relative speeds of disks are differentiated by the number of *broadcast units*[11] on the disks. Data located on a disk with less broadcast units is scheduled for broadcast more frequently than a disk with many broadcast units. The relative speeds and broadcast frequency of broadcast disks inversely proportional

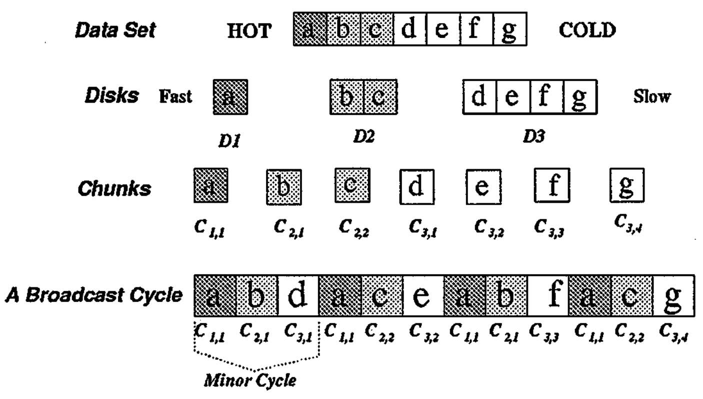

Figure 13. An example of a seven-page, three-disk broadcast program.

to the number of broadcast units on those disks. Thus, data frames with higher demands usually are placed on a higher speed broadcast disk.

The broadcast units on broadcast disks, called *chunks*, have equal size.[12] The broadcast schedule is generated by broadcasting a chunk from each disk and cycling through all the chunks sequentially over all the disks. A *minor cycle* is defined as a sub-cycle consisting of one chunk from each disk. Consequently, chunks in a minor cycle are repeated only once and the number of minor cycles in the broadcast equals the *least common multiple* (LCM) of the relative frequency.

Unlike traditional disks where the number and the capacity of the disks are fixed by hardware, The broadcast disks has flexibility in deciding the number, the size, the relative spinning speed, and the data frame placement of each disk. Broadcast schedules can be programmed once the data frames, relative speed of each disk, the number of data frames placed on the disks, and the size of each disk are determined.

Figure 13 illustrates an example where seven chunks are divided into three ranges of similar average access probabilities [1]. Each of which will be assigned to a separate disk in the broadcast. In the figure, chunk, $C_{i,j}$, refers to the jth chunk of disk i. Chunks in the first disk are to be broadcast twice as frequently as chunks in the second one and four times as often as those of the slowest disk.

However, the reason that multi-disk broadcast can achieve better performance than a random broadcast schedule and the expected access time for retrieving data frames from the broadcast disks were not given in [1]. In the following, we prove Theorem 1 used in the paper.

Proof: The data frame d is scheduled to broadcast f_i times in a cycle, the length of the broadcast (in number of frames) is D:

$$D = \sum_{j=1}^{N} F_j \cdot f_j$$

Assuming the inter-arrival time for each broadcast of d is D_j (where $1 \leq j \leq f_i$). D_j is the number of frames between two consecutive copies of d. k is the number of frames between the frame where clients begin monitoring channels and the next copy of d. Therefore, the expected access time for d can be estimated as:

$$\frac{1}{D} \cdot \sum_{j=1}^{f_i} \sum_{k=0}^{D_j} \left(\frac{1}{2} + D_j - k \right) = \frac{1}{D} \cdot \sum_{j=1}^{f_i} \frac{(D_j + 1)^2}{2}$$

Since $\sum_{j=1}^{f_i} D_j = D - f_i$ is a constant, when $D_j = D_{j+1}$ for $1 \leq j \leq f_i - 1$, $\sum_{j=1}^{f_i} D_j^2$ will have a minimum value. Note that $D_j + 1$ is the number of frames from one broadcast of d to the next broadcast of d. If all the f_i broadcasts of d are equally spaced, then we have $D_j + 1 = D/f_i$. As a result, the minimum expected number of data frames retrieved before the desired one arrives for data on disk with frequency f_i, denoted as C, can be expressed as follows.

$$C = \frac{D}{2 f_i} \tag{9}$$

□

Notes

1. In this paper, we use ‘client’ or ‘mobile client’ to refer to a user with a mobile computer.
2. Every indexing technique usually introduces non-zero access time overhead.
3. In this paper, we only consider the case of single attribute indexing and clustering. Issues involving multiple attribute indexing and clustering are addressed in [10].
4. Even so, the integrated signature and multi-level signature schemes can benefit from a clustered data organization for broadcast channels.
5. To simplify our discussion, we neglect the variance of the meta segment size.
6. This theorem is consistent with similar formulae presented in a separate research approach [19].
7. The broadcast frequency is used to represent the speed of a disk.
8. For simplicity, the three pointers of each index node in the lower most index tree level is represented by just one arrow.
9. Note that it is different from the meta segments for a non-clustered flat broadcast, where frames with the same attribute value may be scattered in several meta segments.
10. Each data frame may have different false drop probabilities. To simplify the cost model, we use average false drop probability to estimate the access time and the tune-in time when a large number of queries are sampled (i.e., many data frames are retrieved).
11. A broadcast unit may consist of one or many data frames, e.g., a cluster of data frames with the same attribute value.
12. In real implementation, chunks can be replaced by variable-sized data frames or a group of data frames.

References

1. S. Acharya, R. Alonso, M. Franklin, and S. Zdonik, “Broadcast disks: Data management for asymmetric communications environments,” in Proceedings of the ACM SIGMOD Conference on Management of Data, San Jose, California, May 1995, pp. 199–210.

2. S. Acharya, M. Franklin, and S. Zdonik, “Dissemination updates on broadcast disks,” in Proceedings of the 22rd VLDB Conference, Mumbai (Bombay), India, Sept. 1996, pp. 354–365.
3. S. Acharya, M. Franklin, and S. Zdonik, “Prefetching from a broadcast disk,” in Proceedings of the International Conference on Data Engineering, New Orleans, LA, Feb. 1996.
4. S. Acharya, M. Franklin, and S. Zdonik, “Balancing push and pull for data broadcast,” in Proceedings of the ACM SIGMOD Conference on Management of Data, Tuscon, Arizona, May 1997, pp. 183–194.
5. D. Barbara and T. Imielinksi, “Sleepers and workaholics: Caching strategies for mobile environments,” in Proceedings of the ACM SIGMOD Conference on Management of Data, Minneapolis, Minnesota, May 1994, pp. 1–12.
6. M.-S. Chen, P.S. Yu, and K.-L. Wu, “Indexed sequential data broadcasting in wireless mobile computing,” in The 17th International Conference on Distributed Computing Systems (ICDCS97), Baltimore, Maryland, USA, May 27–30, 1997.
7. S. Hameed and N.H. Vaidya, “Efficient algorithms for a scheduling single and multiple channel data broadcast,” Technical Report 97-002, Computer Science, Texas A&M University, Feb. 1997.
8. Q.L. Hu, D.L. Lee, and W.-C. Lee, “A comparison of indexing methods for data broadcast on the air,” in Proceedings of the 12th International Conference on Information Networking (ICOIN–12), Jan. 1998, pp. 656–659.
9. Q.L. Hu, D.L. Lee, and W.-C. Lee, “Optimal channel allocation for data dissemination in mobile computing environments,” in Proceedings of the 18th International Conference on Distributed Computing Systems (ICDCS98), Amsterdam, The Netherlands, May 1998, pp. 480–487.
10. Q.L. Hu, W.-C. Lee, and D.L. Lee, “Indexing techniques for power management in multi-attribute data broadcast,” ACM/Baltzer Mobile Networking and Applications (MONET), to appear.
11. T. Imielinski, S. Viswanathan, and B.R. Badrinath, “Energy efficiency indexing on air,” in Proceedings of the International Conference on SIGMOD, 1994, pp. 25–36.
12. T. Imielinski, S. Viswanathan, and B.R. Badrinath, “Power efficiency filtering of data on air,” in Proceedings of the International Conference on Extending Database Technology, 1994, pp. 245–258.
13. T. Imielinski, S. Viswanathan, and B.R. Badrinath, “Data on the air—organization and access,” IEEE Transactions of Data and Knowledge Engineering, vol. 9, no. 3, 1997, pp. 353–372.
14. W.-C. Lee, Q.L. Hu, and D.L. Lee, “A study of channel allocation methods for data dissemination in mobile computing environments,” ACM/Baltzer Mobile Networks and Applications (MONET), vol. 4, no. 2, pp. 117–129, 1999.
15. W.-C. Lee and D.L. Lee, “Using signature techniques for information filtering in wireless and mobile environments,” Special Issue on Database and Mobile Computing, Journal on Distributed and Parallel Databases, vol. 4, no. 3, pp. 205–227, 1996.
16. W.-C. Lee and D.L. Lee, “Signature caching techniques for information filtering in mobile environments” ACM Wireless Networks, vol. 5, no. 1, pp. 57–67, 1999.
17. N. Shivakumar and S. Venkatasubramanian, “Efficient indexing for broadcast based wireless systems,” ACM/Baltzer Mobile Networks and Applications (MONET), vol. 1, no. 4, pp. 433–446, 1996.
18. K. Stathatos, N. Roussopoulos, and J.S. Baras, “Adaptive data broadcast in hybrid networks,” in Proceedings of the 23rd VLDB Conference, Athens, Greece, Aug. 1997, pp. 326–335.
19. N.H. Vaidya and S. Hameed, “Scheduling data broadcast in asymmetric communication environments,” Technical Report 96-022, Computer Science, Texas A&M University, Nov. 1996.
20. S. Zdonik, M. Franklin, and S. Acharya, “Are disks in the air’ just pie in the sky?” in IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, Dec. 1994.