Mobile Networks and Applications 6, 185–197, 2001

Indexing Techniques for Power Management in Multi-Attribute Data Broadcast

QINGLONG HU *
Department of Computer Science, University of Science and Technology, Hong Kong

WANG-CHIEN LEE
GTE Laboratories Incorporated, Waltham, MA 02451, USA

DIK LUN LEE
Department of Computer Science, University of Science and Technology, Hong Kong

Abstract. In this paper, we discuss the power conservative indexing techniques for managing multi-attribute data broadcast on wireless channels. These indexing techniques, namely, *index tree*, *signature* and *hybrid*, aim at improving the battery power consumption of mobile clients. By taking into account the broadcast management factors such as *clustering* and *scheduling*, these three indexing schemes may significantly reduce tune-in time while maintaining a reasonable access time. Cost models for single and multi-attribute query processing are developed. Our performance evaluation shows that the signature and hybrid methods are superior to the index tree method.

Keywords: indexing, multi-attribute query, power conservation, data broadcast, mobile computing

1. Introduction

Wireless communication and mobile computing have grabbed a lot of attention from the computer and communication research communities. Among the various issues under study, efficiency management of the scarce wireless bandwidth and limited battery power is critical to the realization of mobile systems. Wireless data broadcast is an attractive and important service for data dissemination in the mobile environments [7], because it allows simultaneous information access by an arbitrary number of users, and thus facilitates efficient bandwidth usage. Meanwhile, mobile computers accessing data by monitoring broadcast channels consume less battery power than those accessing data by sending requests through traditional point-to-point communications. A lot of studies on data broadcast have appeared in the literature [1,7,13,16].

Power conservation techniques for the management of wireless data broadcast services include indexing, clustering and scheduling techniques. Among the proposed indexing techniques, the signature [13,14] and the index-tree [9] methods represent two major classes[1] of power conserving indexing techniques for wireless data broadcast. Evaluation has been conducted on each of the indexing technique. A comparative study on these two index methods was conducted for clustered broadcast cycle [5]. Moreover, in a recent paper [6], we further examined these two indexing techniques by considering clustering and scheduling, and proposed a hybrid indexing method which combined the strength of the signature and index-tree methods. However, all of the above techniques for data broadcast management are focused on single-attribute based data organizations and queries. In the real world applications, data items usually contain multiple attributes. Moreover, complex multi-attribute queries are frequently issued for various business applications. Thus, it is important to design wireless broadcast services for multi-attribute data set and to support both of single and multi-attribute queries.

In this paper, we investigate the issues and methods for wireless data broadcast services which facilitate multi-attribute data access. We study three different power conserving indexing techniques, namely *index tree*, *signature*, and *hybrid*, and their supporting data organizations for broadcast of multi-attribute data items. These three techniques are index-based, but the factors of data clustering and broadcast scheduling have also been taken into consideration. For each method, query processing algorithms for both of single and multi-attribute queries are proposed. We develop cost models and evaluate these two types of queries in terms of access and power efficiencies. To demonstrate our research result, analytical comparisons of the query performance based on various sizes of broadcast cycle are conducted.

The rest of the paper is organized as follows. In section 2, we give an informal introduction of the wireless data broadcast services. In section 3, power conserving indexing techniques and access methods for multi-attribute queries are introduced. In section 4, we evaluate the power conserving indexing techniques and their query efficiency for both of the single and multi-attribute queries. Finally, section 5 concludes the paper.

* Now with Department of Computer Science, University of Waterloo, Canada.

[1] In [10], hashing and flexible indexing methods were explored.

2. Multi-attribute data broadcast service

The application of wireless data broadcast is enormous, including ubiquitous access of stock ticks, traffic and weather information. In this paper, we consider the wireless data broadcast services for users equipped with battery powered mobile computers. Among the various issues faced by the wireless data broadcast services, we are especially interested in the data broadcast management issues concerning power conservation.

A wireless data broadcast service involves two parties: the base station and the mobile clients. The base station serves the role of the data server which maintains a set of records[2], including formatted and multimedia data such as text, image, audio and video. To support wireless data broadcast services, the base station periodically delivers these records to its clients as a series of *data frames* through one *shared* broadcast channel. A data frame, the logical unit of information broadcasted on the air, consists of *packets*[3] which are the physical units of broadcast. A frame contains a header for synchronization and meta-information that indicates the type and length of the frame. Since data frames are periodically broadcast, a complete broadcast of data frames is called a *broadcast cycle*. From the user's viewpoint, broadcast data are perceived as a stream of frames flowing along the time axis. Logically, there is no specific start and end frame for a broadcast cycle. A broadcast cycle starts with any frame and ends when the frame appears again. Updates to the frames are reflected between successive broadcast cycles. To receive the frames from the air, mobile applications or users issue queries which specify values for one or multiple common attributes to their mobile computers. As a result, the mobile computers monitor the broadcast channels and return the frames that satisfy these queries.

There are two factors affecting power consumption in mobile computers: (1) mobile computers can be switched between *active* mode (full power) and *doze* mode, and (2) receiving messages consumes less energy than sending messages. Methods have been proposed for energy efficient query optimization [2], caching [14,19] and information broadcasting [8,13,15].

Three major techniques can be used for power conservation of the wireless data broadcast services: indexing, scheduling and clustering techniques. Indexing technique is based on the idea that by interleaving auxiliary information with data frames on broadcast channels, mobile computers are able to predict the arrival of requested data frames. Hence, by staying in doze mode most of the time and only waking up to receive the data when they arrive (also called *selective tune-in*), power consumption of the mobile clients can be conserved.

[2] To facilitate our discussion, we assume that each record consists of a number of common attributes.

[3] In our analysis, we assume that the size of a data frame is a multiple of the packet size.

In addition to the secondary indexing mechanisms, broadcast data organizations and data access methods play important roles in determining the access and power efficiencies of mobile computers. In the following, we discuss the factor of scheduling and clustering considered in designing broadcast data organization for the power conserving facilities we employeed.

Broadcast scheduling determines the contents of the broadcast cycle and the broadcast frequency of the data frames [1,4,12,16,17] (i.e., the hot data set for broadcast and the relative broadcast frequency of each frame). A simple broadcast schedule, called *flat broadcast*, is to broadcast each data frame once every cycle. Since the client access pattern for an attribute is usually skewed (i.e., some data are accessed more frequently than others), another scheduling method, called *broadcast disks*, was proposed in [1]. Broadcast disks broadcast important data more often than the other data to reduce the average response time for queries based on that attribute value. However, broadcast disks increase the length of a broadcast cycle and the client has to spend more time retrieving the less commonly interested data. For multi-attribute data records, a cycle can be organized in broadcast disks based on one of attributes. Due to data frame duplication, queries other than that attribute may take longer to answer. Therefore, broadcast disks are not efficient for data records with multiple attributes. Thus, in this paper, flat broadcast scheduling is used.

Data clustering refers to arranging the data items with the same value for a specific attribute to appear consecutively in a broadcast cycle [6,8,10]. By monitoring the arrival of the first data item with the desired attribute value, the client can retrieve all of the successive data items with the same attribute value instead of filtering for each arrival of the desired data items. However, similar to broadcast scheduling, a broadcast cycle can only be clustered based on one attribute. Although the other attributes are not clustered in the cycle, a second attribute can be chosen to cluster the data items within a data cluster of the first attribute. In turn, a third attribute can be chosen to cluster the data items within a data cluster of the second attribute. The same process may be applied for further clustering based on other attributes. We call the first attribute the *clustered attribute* and the other attributes the *non-clustered attributes*.

For each non-clustered attribute, the broadcast cycle can be partitioned into a number of segments called *meta segments* [11], so that each of which holds a sequence of frames with non-decreasing (or non-increasing) values of that attribute. Thus, when we look at each individual meta segment, the data frames are clustered on that attribute and indexing techniques designed for clustered data broadcast can still be applied within the meta segment. To facilitate our study, we define the number of meta segments in the broadcast cycle for an attribute as the *scattering factor* of the attribute. For multiple attributes, the broadcast cycle is partitioned into meta segments for each attribute in the order that frequently accessed attributes first. The scattering factors of an attribute increases as the importance of the attribute de-

creases. Organizing data broadcast with the above discussed clustering structure can improve retrieval efficiency. Thus, in this paper, we assume that the data frames within a broadcast cycle are partitioned into meta segments based on the first r attributes.

By taking the clustering and scheduling factors into consideration, we discuss three index-based power conservation techniques for wireless data broadcast services. Based on various system and applications parameters (such as the size of the broadcast data set), different power conservation techniques may be employeed for efficient broadcast management.

To facilitate our evaluation of the power conservation techniques, the following criteria are used to estimate the efficiency of the techniques based on their support for multi-attribute query processing:

- *Access time*: the average time elapsed from the moment a client makes a query to the moment when all the requested data frames are received by the client.
- *Tune-in time*: the time spent by a mobile computer staying active in order to acquire the requested data.
- *Indexing efficiency*: the tune-in time saved versus the access time overhead introduced by indexing.

While access time measures the efficiency of access methods and data organizations for broadcast channels, tune-in time is frequently used to estimate the power consumption by a mobile computer. The indexing techniques can reduce tune-in time, but they usually lengthen a broadcast cycle and hence increase the access time (called *index overhead*). 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 conserving indexing technique has to balance out the overhead on access time and the saved tune-in time in order to maximize the indexing efficiency. Access time, tune-in time, and indexing efficiency are measured in terms of number of packets.

3. Power conserving indexing techniques

In this section, we investigate the application of the index tree, the signature and the hybrid techniques for power conservation in wireless data broadcast services. Multiple attributes data set is considered in the services. Two types of queries, *single-attribute queries* and *multi-attribute queries*, are used for the evaluation of these multi-attribute indexing techniques. The cost models of access time and tune-in time for the investigated indexing techniques are derived. The comparisons among these three techniques based on access time, tune-in time and indexing efficiency are presented in the next section. To save space, we do not repeat the basic index tree, signature and hybrid techniques in the paper. Readers please refer to [6,11,13] for details.

Table 1
System parameter setting.

Parameter	Description
F	number of data frames in a broadcast cycle
P	average number of packets in a data frame
m	number of attributes in a frame
r	number of indexed attributes in a frame ($0 \leqslant r \leqslant m$)
q	number of attributes in a query ($1 \leqslant q \leqslant m$)
S_i	a_i selectivity: percentage of frames with the required attribute value
p_i	probability of queries based on a_i

Before we jump into the indexing techniques and their associated access methods for queries, we first describe the system parameters used in the study. We assume that there are m common attributes in each data frame and the attributes are sorted based on their access frequencies. Let the ordered attributes be $a_1, a_2, \ldots, a_m$ and the access probabilities for the corresponding single-attribute based queries be $p_1, p_2, \ldots, p_m$, where $p_i \geqslant p_{i+1}$ $(1 \leqslant i < m)$. a_1, called the *major attribute*, is the most frequently accessed attribute and the rest of the attributes are called the *minor attributes*. The minor attribute a_m is the least accessed attribute. Indexes are built on multiple attributes (i.e., the frequently accessed attributes, $a_1, a_2, \ldots, a_r$ where $1 \leqslant r \leqslant m$). The rest system parameters are defined in table 1.

For the single-attribute based queries, a query is involved with only one attribute. Thus, $\sum_{i=1}^{m} p_i = 1$. We choose to estimate the cost of single-attribute queries by computing the average cost for querying every attribute (i.e., $a_1, \ldots, a_m$), separately. For multi-attribute queries, a query may contain multiple attributes and consist of many combinations of Boolean operators, such as conjunction ($\wedge$) and disjunction ($\vee$). Since it is impossible to consider all combinations, we choose the queries with either all conjunction or all disjunction operators as the representative multi-attribute queries for evaluation.

Let $Q\{a'_1, \ldots, a'_q\}$ denote a query involved with q attributes, where the attributes a'_i $(1 \leqslant i \leqslant q \leqslant m)$ are sorted according to the order they are indexed. a'_1 may be the major or the minor attribute and a'_i $(2 \leqslant i \leqslant q)$ are the minor attributes. To simplify our discussion, we develop the cost models for multi-attribute queries with three attributes (including one major attribute and two minor attributes denoted as $Q\{a_1, a_i, a_j\}$, where $1 < i < j \leqslant r$). Hence, $Q\{a'_1 \wedge a'_i \wedge a'_j\}$ and $Q\{a'_1 \vee a'_i \vee a'_j\}$ are the only two query expressions. This is the minimal configuration to cover the common queries involved.

To provide the comparison baseline, we first derive formulae for a wireless data broadcast service with no index (denoted *non-index*). For the rest of the paper, we use A and T in our formulae to denote access time and tune-in time, respectively. Meanwhile, we use idx, sig and hyb in superscripts to denote their corresponding index methods.

For the non-index broadcast, since every arriving frame must be retrieved into the client cache to check against the attribute values specified in the query, the tune-in time is

very long and is equal to the access time. For a clustered major attribute, a_1, the estimated access time and tune-in time for $Q\{a_1\}$ and $Q\{a_1 \wedge a_i \wedge a_j\}$ is as follows:

$$A(a_1) = T(a_1) = A(a_1 \wedge a_i \wedge a_j) = T(a_1 \wedge a_i \wedge a_j) = PF\left(\frac{1}{2} + S_1\right).$$

For a nonclustered attribute a_k, the estimated access time and tune-in time for $Q\{a_k\}$ and $Q\{a_1 \vee a_i \vee a_j\}$ is as follows:

$$A(a_k) = T(a_k) = A(a_1 \vee a_i \vee a_j) = T(a_1 \vee a_i \vee a_j) = PF,$$

since the client must scan the entire broadcast cycle to retrieve the desired frames.

3.1. The index tree method

For a multi-attribute data set, we can build index trees for each attribute separately. Several factors influence the order of attribute indexes in a cycle. To reduce index tree overhead for multiple attributes, the index tree should be built in the order that the attribute with the highest selectivity is indexed first. In this way, the clustered data frames can be received successively. On the other hand, to achieve the best average system performance, index should be built in the order of access frequencies (i.e., the higher the access frequency the higher priority an attribute should be chosen for indexing). From the index point of view, an index built on an attribute with low selectivity is better (i.e., it has high partitioning power and consequently, high filtering capability). Obviously, the factors of selectivity and access probability have contradicting effects. We have to balance the access probability and the attribute selectivity to determine the order of attributes to be indexed.

Based on research results in [6,11], data access is more efficient for clustered attribute than for non-clustered attribute (i.e., both access time and tune-in time are improved by clustering). In addition, the index built on clustered attribute requires less overhead than built on non-clustered attributes. Since data frames can be clustered on at most one attribute, we choose to cluster data frames in a broadcast cycle based on major attribute (i.e., a_1). Thus, the minor attributes are non-clustered. Let $I = \{a_1, \ldots, a_r\}$ be the set of indexed attributes. The broadcast cycle is partitioned separately for each attribute in I ordered from a_1 to a_r. Hence, the scattering factor for a_i, denoted as M_i, increases with i. The index tree overhead of a_i depends on M_i. The smaller the subscript of the attribute in I, the lower the index overhead.

Since the broadcast channel is a linear medium and the index information increases the length of the cycle, index for an attribute influences the performance of queries based on not only that attribute but other attributes as well. When M_i (where $i > r$) is too large, constructing a distributed indexing tree on a_i does not help improve tune-in time much but results in a high index overhead (i.e., the broadcast cycle can become very long and the access time for queries based on any attribute increases considerably). In this case, non-index is a better approach.

Table 2
Parameter setting for index tree.

Parameter	Description
h_i	height of the whole index tree built for a_i
t_i	number of upper levels in the index tree for a_i which are replicated
N	number of packets in an index tree node
n	number of search-key plus pointers that a node can hold

The value of M_j depends on S_i and the inter-relation between a_i and a_j, where $i < j$. However, the relationship among the attributes is difficult to obtain since it is semanticly based and varies from application to application. To simplify the cost model, we assume that the attributes are *random and independent*. Based on this assumption, we can develop a simple estimation on M_i:

$$M_i = \frac{1}{\prod_{j=1}^{i-1} S_j}, \tag{1}$$

where $M_1 = 1$.

Table 2 describes the parameter settings for index tree cost models. In addition, we use $X[h]$ and $X[t]$ to denote the total number of nodes in the full index tree and the replicated part of the index tree, respectively. The number of nodes in the ith level of the index tree be denoted as $L[i]$. For multi-attribute indexing, extra index information increases the cycle length to $PF + N\sum_{i=1}^{r} E_i$ packets, where E_i is the total number of index nodes allocated to a_i. For a_i, $i \neq 1$, there is an index tree constructed for each meta segment. Therefore, there are M_i index trees corresponding to a_i. Since each index tree has $X[h_i] + L[t_i + 1] - 1$ index nodes allocated for a_i within a meta segment, the total overhead to build index trees for a_i in a broadcast cycle is $E_i = M_i(X[h_i] + L[t_i + 1] - 1)$ nodes. On the average, each data frame has $TREE = N\sum_{i=1}^{r} E_i/F$ extra packets due to multi-attribute indexing. The replicated index tree part is broadcast every $1/L[t_i + 1]$ of each meta segment. The data frames are divided into $M_i L[t_i + 1]$ data segments with replicated index nodes at the beginning of each data segment. The length of each data segment is $(PF + N\sum_{i=1}^{r} E_i)/(M_i L[t_i + 1])$. Hence, the *initial probe time*, the time for a mobile client to reach the index tree built for a_i, can be estimated [9] as follows:

$$PROBE^{\text{idx}} = \frac{PF + N\sum_{i=1}^{r} E_i}{2M_i L[t_i + 1]}.$$

If the index tree is fully balanced, for $1 \leqslant i \leqslant r$, the height of the tree for a_i, h_i, the number of nodes on the jth level of the index tree, $L[j]$, and the number of nodes in the upper t levels of the index tree, $X[t]$, can be calculated based on [6,11].

3.1.1. Index tree method for single-attribute queries

The access method for single-attribute queries in a multi-attribute data broadcast service using the index tree tech-

nique for power conservation is as follows. The mobile client tunes into the broadcast channel to locate the next index tree built based on the query attribute. Then, by following the links in the index tree, the arrival of the desired data items may be obtained.

For $Q\{a_1\}$, the waiting time for the desired data to arrive is half of a broadcast cycle. The retrieval time is equal to PS_1F. Therefore, the estimation for access time is

$$\begin{aligned} A^{\text{idx}}(a_1) &= \text{initial probe time} \\ &\quad + \text{waiting time before first desired frame arrives} \\ &\quad + \text{retrieval time for all desired frames} \\ &\quad\quad \text{in the broadcast} \\ &= PROBE^{\text{idx}} + \frac{PF + N\sum_{i=1}^{r} E_i}{2} + PS_1F. \end{aligned}$$

For $Q\{a_i\}$ where $1 < i \leqslant r$, the sum of the waiting time and the retrieval time is the total length of the broadcast. Hence, the access time is

$$\begin{aligned} A^{\text{idx}}(a_i) &= \text{initial probe time} \\ &\quad + \text{retrieval time for all desired frames} \\ &\quad\quad \text{in the broadcast} \\ &= PROBE^{\text{idx}} + PF + N\sum_{i=1}^{r} E_i. \end{aligned} \tag{2}$$

The tune-in time for $Q\{a_i\}$ where $1 \leqslant i \leqslant r$ depends on the initial probe, the scanning of index tree, the extra scanning of index trees in subsequent meta segments, and the retrieval of S_iF data frames. Thus, it is bound by

$$T^{\text{idx}}(a_i) = (h_i + 1)N + (M_i + S_iF)P.$$

For $Q\{a_i\}$ where $r < i \leqslant m$, there is no index to help the client to make selective tuning. We assume that the client needs to retrieve one tree node to get each index tree arrival information so that the client can skip the index trees. The access time and tune-in time are:

$$\begin{aligned} A^{\text{idx}}(a_i) &= N\sum_{i=1}^{r} E_i + PF, \\ T^{\text{idx}}(a_i) &= rN + PF. \end{aligned}$$

Therefore, the average access time and tune-in time for querying all m attributes can be estimated as

$$\begin{aligned} ACCESS^{\text{idx}} &= p_1A^{\text{idx}}(a_1) + \sum_{i=2}^{r} p_iA^{\text{idx}}(a_i) \\ &\quad + \sum_{i=r+1}^{m} p_iA^{\text{idx}}(a_i), \\ TUNE^{\text{idx}} &= p_1T^{\text{idx}}(a_1) + \sum_{i=2}^{r} p_iT^{\text{idx}}(a_i) \\ &\quad + \sum_{i=r+1}^{m} p_iT^{\text{idx}}(a_i). \end{aligned}$$

3.1.2. Index tree method for multi-attribute queries

In this section, we first develop the cost formulae for queries with conjunction operator ($\wedge$), then for queries with disjunction operator ($\vee$). The general access method for $Q\{a'_1 \wedge a'_2 \wedge \cdots \wedge a'_q\}$ is:

Search. For each query attribute a'_i DO

- initial probe for the index tree built based on a'_i within the data segment qualified for a'_{i-1},
- search the index tree built based on a'_i: the client follows a list of pointers to find out the arrival time of the desired data frame.

Retrieval. Scan the current meta segment for the desired data frames. At the end of the meta segment, a jump is made to the other meta segments where the client should examine whether that meta segment is qualified for the attributes $a'_1, \ldots, a'_q$ contained in the query.

The access time for the query $Q\{a_1 \wedge a_i \wedge a_j\}$ is

$$\begin{aligned} &A^{\text{idx}}(a_1 \wedge a_i \wedge a_j) \\ &\quad = \text{initial probe time for major attribute index} \\ &\quad\quad + \text{waiting time for the first desired data segment} \\ &\quad\quad + \text{access time for desired data frames} \\ &\quad = PROBE^{\text{idx}} + \frac{PF + N\sum_{l=1}^{r} E_l}{2} + DS(a_i, a_j), \end{aligned}$$

where:

- if $r < i < j$ (i.e., both a_i and a_j are not indexed),
$$DS(a_i, a_j) = SPF,$$
- if $i \leqslant r$ and $j > r$ (i.e., only a_i is indexed),
$$DS(a_i, a_j) = \frac{(P + N\sum_{l=1}^{r} E_l/F)S_1F}{2} + \frac{PS_iF}{M_i},$$
- if $i < j < r$ (i.e., both a_i and a_j are indexed),
$$DS(a_i, a_j) = \frac{(P + N\sum_{l=1}^{r} E_l/F)S_1F(1 + S_i)}{2} + \frac{PS_jF}{M_j}.$$

The tune-in time for $Q\{a_1 \wedge a_i \wedge a_j\}$ depends on the number of levels of the index trees for a_1, a_i, and a_j and the selectivity of the conjunction query. The initial probe of the client is to find the occurrence of the control index for a_1. The control index directs the client to the required higher level index tree nodes for a_1. A search in the index tree is conducted and the last pointer in the index tree points to the occurrence of the desired data segment. Within that data segment, a probe for index tree for a_2 is conducted. Once the index tree for a_2 is obtained, a search in the index tree is carried out to find the data segment which matches a_2. Similarly, the filtering process is continued with the rest of attributes involved with the query. Finally, the data frames of interest are received. As a result, the tune-in time is obtained as follows:

Table 3
Parameter setting for signature scheme.

Parameter	Description
P_i	a frame false drop probability for queries based on a_i
$P_{\mathrm{f}}^{\mathrm{S}}$	a simple signature false drop probability
$P_{\mathrm{f}}^{\mathrm{I}}$	an integrated signature false drop probability
k	number of data frames indexed by an integrated signature
l_i	average number of true matches in a frame group for a_i
p	number of bits in a packet
R_{S}	the size (number of packets) of a simple signature
R_{I}	the size (number of packets) of an integrated signature
s	number of bit strings which are superimposed into a signature

- if $r < i < j$ (i.e., both a_i and a_j are not indexed),

$$T^{\mathrm{idx}}(a_1 \wedge a_i \wedge a_j) = (1 + h_1)N + (1 + S_1 F)P,$$

- if $i \leqslant r$ and $j > r$ (i.e., only a_i is indexed),

$$T^{\mathrm{idx}}(a_1 \wedge a_i \wedge a_j) = (1 + h_1 + h_i)N + (1 + S_1 S_i F)P,$$

- if $i < j < r$ (i.e., both a_i and a_j are indexed),

$$\begin{aligned} &T^{\mathrm{idx}}(a_1 \wedge a_i \wedge a_j) \\ &\quad = (1 + h_1 + h_i + h_j)N + (1 + S_1 S_i S_j F)P. \end{aligned}$$

For disjunctive queries, $Q\{a'_1 \vee a'_2 \vee \cdots \vee a'_q\}$, the client monitors the channel for every index tree built on the attributes in the query, just as the client queries each of q attributes simultaneously (i.e., within one scan). The following is the access method:

Initial probe. For any index trees built based on $a'_1, \ldots, a'_q$, the client tunes into the broadcast channel and determines when the next index tree nodes for the data frames are broadcast.

Search. For each index tree built based on $a'_1, \ldots, a'_q$, the client follows a list of pointers to find out the arrival time of the desired data frame.

Retrieval. The client tunes into the broadcast channel to download all the qualified frames.

The access time for $Q\{a_1 \vee a_i \vee a_j\}$ is similar to that for $Q\{a_i\}$ where $1 < i \leqslant r$ and is given in equation (2). The tune-in time for $Q\{a_1 \vee a_i \vee a_j\}$ is as follows:

- if $r < i$ or $r < j$ (i.e., if a_i or a_j are not indexed),

$$T^{\mathrm{idx}}(a_1 \vee a_i \vee a_j) = rN + PF,$$

- if $i < j < r$ (i.e., both of a_i and a_j are indexed),

$$\begin{aligned} &T^{\mathrm{idx}}(a_1 \vee a_i \vee a_j) \\ &\quad = (1 + h_1 + h_i + h_j)N + (M_1 + M_i + M_j)P \\ &\qquad + (S_1 + S_i + S_j - S_1 S_i - S_1 S_j \\ &\qquad\quad - S_i S_j + S_1 S_i S_j)FP. \end{aligned}$$

3.2. The signature methods

A signature of a data frame is basically a bit vector generated by first hashing the attribute values in the data frame into bit strings and then superimposing them together. Signatures are broadcast along with their associated data frames in a broadcast cycle. By checking a signature, the mobile computer can decide whether a data frame contains the desired information. If it does not, the mobile computer turns into doze mode and wakes up again for the next signature. There are several signature methods, namely, simple, integrated and multi-level signature methods [13,14]. 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. Based on experiences accumulated in previous study [6], we choose the multi-level signature method for multi-attribute data broadcast services.

Table 3 defines the parameters for signature cost models. The estimation of the signature false drop probability for either simple signature or integrated signature is given as follows [13]:

Lemma 1 (Optimal false drop probability). Given the number of packets in a signature, R hereafter denotes either R_{S} or R_{I}, the number of bit strings superimposed into the signature, s, the false drop probability for the signature is: $P_{\mathrm{f}}^{\mathrm{S}} = P_{\mathrm{f}}^{\mathrm{I}} = 2^{-R(p \ln 2)/s}$.

We define P_i as the false drop probability of a data frame for queries based on a_i. P_i has two components $P_{\mathrm{f}}^{\mathrm{I}}$ and $P_{\mathrm{f}}^{\mathrm{S}}$, which, respectively, reflect the false drops introduced by the integrated signature and the simple signature. Based on [13], we derive the estimation of P_i as follows:

$$P_i = \frac{\left(\frac{1}{k} - \left\lceil \frac{S_i}{l_i} \right\rceil\right) P_{\mathrm{f}}^{\mathrm{I}} \left(\frac{kR_{\mathrm{S}}}{P} + kP_{\mathrm{f}}^{\mathrm{S}}\right) + \left\lceil \frac{S_i}{l_i} \right\rceil (k - l_i) P_{\mathrm{f}}^{\mathrm{S}}}{(1 - S_i)}. \quad (3)$$

To simplify our discussion, we assume that frames with the same attribute value for a_i ($1 \leqslant i \leqslant m$) are evenly distributed in each meta segment. Consequently, the average number of the frames with the same value for the attribute a_i in each meta segment is $\lceil S_i F / M_i \rceil$, where M_i can be estimated by equation (1). Let k be the number of data frames indexed by an integrated signature. The number of distinct attribute values used for signature generation, s, can be estimated as $\lceil k / \lceil S_i F / M_i \rceil \rceil$. For frames in a meta segment partitioned for a_i, the average number of qualified frames corresponding to a matched integrated signature, l_i ($1 \leqslant l_i \leqslant k$ for $1 \leqslant i \leqslant r$), called *locality of true matches*, can be estimated as $l_i = k / \lceil k / \lceil S_i F / M_i \rceil \rceil$. For randomly distributed frames, the locality of true matches corresponding to a_i ($r < i \leqslant m$), l_i, is equal to 1. Since each data frame contains m attribute values[4] corresponding to $a_1, \ldots, a_m$, the expected distinct superimposed bit strings, s, for an integrated signature can be estimated as $s = \sum_{i=1}^{m} \lceil k / l_i \rceil$, where $\lceil k / l_i \rceil$ is the expected number of distinct bit strings superimposed for attribute a_i.

[4] We assume that the hashed attribute values for different attributes are not the same (i.e., independent event); therefore, the number of distinct bit strings superimposed into a simple signature for m attributes is m.

To simplify the formula for the estimation of the tune-in time, we assume that the integrated signature for the group of the initial probe is a true match. The average waiting time for retrieving one data frame is $SIG_S + SIG_I + P$, where SIG_S and SIG_I, the average simple and integrated signature overheads for a data frame, can be calculated as $SIG_S = R_S$ and $SIG_I = R_I/k$. Thus, $CIRCLE^{sig} = (SIG_I + SIG_S + P)F$.

Generally speaking, the length of a signature (R_S or R_I) is very short compared to an data frame (i.e., $P \gg R$); otherwise both the access time and the tune-in time would be large. Therefore, the time for the client to filter out the partial signature and the partial data frame, called initial probe time, can be be approximated as half of the data frame size, $PROBE^{sig} = P/2$. The tune-in time in the initial probe period is the expected sum of the time when the client is active for filtering a partial simple signature, a partial integrated signature, and a partial data frame. For the similar reason, the active time for filtering a partial signature is negligible and the initial tune-in time can be approximated as $P/2$.

3.2.1. Multi-level signature method for single-attribute queries

The access method for single-attribute queries is straightforward (i.e., the signatures are matched one by one). The access time for queries based on major attribute can be derived as follows:

$$\begin{aligned} &A^{sig}(a_1) \\ &\quad = \text{initial probe time} \\ &\qquad + \text{filtering time for the first desired frame to arrive} \\ &\qquad + \text{retrieving time for all the desired frames} \\ &\qquad\quad \text{in the broadcast} \\ &\quad = \frac{P}{2} + F\frac{SIG_I + SIG_S + P)}{2 + PS_1F}. \end{aligned} \tag{4}$$

The access time for queries based on a non-clustered attribute a_i is

$$\begin{aligned} A^{sig}(a_i) &= \text{initial probe time} + \text{a broadcast cycle} \\ &= \frac{P}{2} + F(SIG_I + SIG_S + P). \end{aligned} \tag{5}$$

The tune-in time for queries with the major attribute a_1 can be approximated as

$$\begin{aligned} &T^{sig}(a_1) \\ &\quad = \text{the tune-in time in the initial probe period} \\ &\qquad + \text{true match data frames in the broadcast} \\ &\qquad + \text{every integrated signature in half the broadcast} \\ &\qquad + \text{simple signatures associated with the qualified} \\ &\qquad\quad \text{integrated signatures} \\ &\qquad + \text{false drop data frames in half the broadcast} \\ &\quad = \frac{P}{2} + PS_1F + F\frac{R_I/k + \lceil S_1/l_1 \rceil k R_S}{2} \\ &\qquad + P_1(F - S_1F)P. \end{aligned} \tag{6}$$

The tune-in time for queries based on a minor attribute a_i is

$$\begin{aligned} &T^{sig}(a_i) = \text{the tune-in time in the initial probe period} \\ &\qquad + \text{true match data frames in the broadcast} \\ &\qquad + \text{every integrated signature in the broadcast} \\ &\qquad + \text{simple signatures associated with the} \\ &\qquad\quad \text{qualified integrated signatures} \\ &\qquad + \text{false drop data frames in the broadcast} \\ &\quad = \frac{P}{2} + PS_iF + F\frac{R_I}{k} + \left(\left\lceil \frac{S_1}{l_1} \right\rceil k R_S\right) \\ &\qquad + P_i(F - S_iF)P. \end{aligned} \tag{7}$$

The overall access time and tune-in time for queries based on an attribute can be estimated as,

$$ACCESS^{sig} = p_1 A^{sig}(a_1) + (1 - p_1)A^{sig}(a_i),$$

$$TUNE^{sig} = p_1 T^{sig}(a_1) + \sum_{i=2}^{m} p_i T^{sig}(a_i).$$

The tune-in time for the multi-level signature is no greater than its corresponding simple signature scheme, because the integrated signature helps the client to avoid retrieving irrelevant data frames with negligible overhead. Since equations (6) and (7) are very complicated, in order to evaluate the optimal signature size to minimize the tune-in time of the multi-level signature, we evaluate the corresponding simple signature instead (i.e., let $P_f^I = k = l_i = 1$ and $R_I = 0$). According to lemma 1, we have $P_f^S \approx 2^{-(p \ln 2/s)R_S}$. We differentiate equation (6) or (7) with respect to R_S and let $\partial TUNE_i/\partial R_S$ equal zero, the optimal simple signature size, $\widehat{R}_S$, can be computed as

$$\widehat{R}_S = \frac{s}{p \ln 2} \log_2 \frac{(1 - S_i)Pp \ln^2 2}{s}. \tag{8}$$

To simplify the signature generation and the match processes on the server and the clients, we usually set both simple and integrated signatures to the same size and equation (8) gives us an approximate method to determine the signature size.

3.2.2. Multi-level signature method for multi-attribute queries

One major difference between the index tree and the signature method is that an index tree node is good for only one indexed attribute, while a signature contains the information for all indexed attributes. Therefore, the client first retrieves the signature into the cache. By scanning that signature, the client knows whether the associated data frame matches the query attributes. Only when the signature matches the query, the corresponding data frame is fetched into the cache.

We develop the access method for queries $Q\{a_1' \wedge a_2' \wedge \cdots \wedge a_q'\}$ as follows. Signature match for a_1' is conducted first. Matched signatures (either true match or false drops) are subjected to further signature matches based on the other query attribute values.

Initial probe. The client tunes into the broadcast channel for the first received signature.

Filtering. For a broadcast cycle, match the signature received with query signatures generated based on attribute values in Q.

IF the signature matches all of query signatures, THEN
IF the signature is an integrated signature, repeat the filtering step for simple signature filtering.
IF the signature is a simple signature, go to the retrieval and check step.
ELSE skip filtering until the next signature of the same level arrives.

Retrieval and check. The client retrieves and further checks frames corresponding to the matched signature in order to eliminate false drops. Repeat filtering step.

For the query $Q\{a_1 \wedge a_i \wedge a_j\}$, the access time is

$$\begin{aligned} A^{\text{sig}}(a_1 \wedge a_i \wedge a_j) &= \text{initial probe time} \\ &\quad + \text{filtering time for the first desired frame to arrive} \\ &\quad + \text{retrieving time for all the desired frames in the broadcast} \\ &= \frac{P}{2} + F\frac{SIG_{\text{I}} + SIG_{\text{S}} + P)}{2} + PS_1F. \end{aligned}$$

The tune-in time for $Q\{a_1 \wedge a_i \wedge a_j\}$ is

$$\begin{aligned} T^{\text{sig}}(a_1 \wedge a_i \wedge a_j) &= \text{the tune-in time in the initial probe period} \\ &\quad + \text{true match data frames in the broadcast} \\ &\quad + \text{every integrated signature in half the broadcast} \\ &\quad + \text{simple signatures associated with the qualified integrated signatures} \\ &\quad + \text{false drop data frames in half the broadcast} \\ &= \frac{P}{2} + PS_1S_iS_jF + F\frac{R_{\text{I}}/k + \lceil S_1/l_1\rceil kR_{\text{S}})}{2} \\ &\quad + PF\big(S_i + \big(S_1 + (1-S_1)P_1\big)(1-S_i)P_i\big) \\ &\quad \times (1-S_j)P_j. \end{aligned}$$

In the above derivation, number of true matches is estimated as $S_1S_iS_jF$ and P_i can be obtained in equation (3).

For a query $Q\{a'_1 \vee a'_2 \vee \cdots \vee a'_q\}$, the access method is as follows:

Initial probe. The client tunes into the broadcast channel for the first received signature.

Filtering. For a broadcast cycle, match the signature received with query signatures generated based on attribute values in Q.

IF the signature matches any of query signatures, THEN
IF the signature is an integrated signature, repeat the filtering step for simple signature filtering.
IF the signature is a simple signature, go to the retrieval and check step.
ELSE skip filtering until the next signature of the same level arrives.

Retrieval and check. The client retrieves and further checks frames corresponding to the matched signature in order to eliminate false drops. Repeat filtering step.

The access time for $Q\{a_1 \vee a_i \vee a_j\}$ is

$$\begin{aligned} A^{\text{sig}}(a_1 \vee a_i \vee a_j) &= \text{initial probe time} + \text{a broadcast cycle} \\ &= \frac{P}{2}2 + F(SIG_{\text{I}} + SIG_{\text{S}} + P). \end{aligned}$$

The tune-in time for $Q\{a_1 \vee a_i \vee a_j\}$ is

$$\begin{aligned} T^{\text{sig}}(a_1 \vee a_i \vee a_j) &= \text{the tune-in time in the initial probe period} \\ &\quad + \text{true match data frames in the broadcast} \\ &\quad + \text{every integrated signature in the broadcast} \\ &\quad + \text{simple signatures associated with the qualified integrated signatures} \\ &\quad + \text{false drop data frames in the broadcast} \\ &= \frac{P}{2} + PF_{\text{t}} + F\left(\frac{R_{\text{I}}}{k} + \left\lceil\frac{S_1}{l_1}\right\rceil kR_{\text{S}}\right) \\ &\quad + P\big(P_1(F - S_1F) + P_i(F - S_iF) + P_j(F - S_jF) \\ &\quad - P_1P_iF(1 - S_1S_i) - P_iP_jF(1 - S_iS_j) \\ &\quad - P_1P_jF(1 - S_1S_j) + P_1P_iP_jF(1 - S_1S_iS_j)\big), \end{aligned}$$

where the number of true matches F_{t} is estimated as $(S_1 + S_i + S_j - S_1S_i - S_1S_j - S_iS_j + S_1S_iS_j)F$.

3.3. The hybrid methods

The hybrid index consists of two parts: sparse index tree and signature. We assume that the multi-level signature is used for the signature part and the sparse index tree is the same as the replicated part of the index tree method. The sparse index tree part does the global filtering based on the major attribute, while the signature part carries out the local filtering based on the attribute values specified in the queries. In this way, the index tree part helps improve tune-in time of the client. The average waiting time for retrieving one data frame from the broadcast cycle can be expressed as

$$TREE + SIG + P = N\frac{X[t_1 + 1] - 1}{F + R}\left(1 + \frac{1}{k}\right) + P,$$

where *TREE* and *SIG* (i.e., $SIG_{\text{I}} + SIG_{\text{S}}$) respectively are the index overheads of the index tree part and the signature part for a frame. The average number of data frames in one data block, $F[B]$, can be calculated in a similar way as in the index tree method, which is $F/L[t_1 + 1]$. Thus, the overheads of the index tree and the signatures in a data block are $F[B]TREE$ and $F[B]SIG$, respectively. Hence, the average initial probe time for the hybrid method and the length of a cycle are

$$PROBE^{\text{hyb}} = \frac{TREE + SIG + P}{2},$$

$$CYCLE^{\text{hyb}} = (TREE + SIG + P)F.$$

3.3.1. The hybrid method for single-attribute queries
For queries based on major attribute a_1[5], the clients first search this sparse index tree to get the arrival time of the data blocks containing the data frames of interest, then signature matching is carried out in the data blocks. The expected access time for queries based on the major attribute is

$$A^{\text{hyb}}(a_1) = (TREE + SIG + P)\frac{1+F}{2} + S_1FP. \quad (9)$$

The tune-in time primarily depends on the initial probe of the client to determine the next occurrence of the sparse index, the access time for the index tree part which equals to the number of levels t_1 of the sparse index tree, the tune-in time for the data block B, and the selectivity S of the major attribute. Therefore, it is bounded by

$$T^{\text{hyb}}(a_1) = (1+t_1)N + T(B) + (1+S_1F)P,$$

where $T(B)$ is the tune-in time for filtering the data block B with the signature method, which can be estimated as

$$\begin{aligned} T(B) = {} & \text{every signature in the length of the data block } B \\ & + \text{false drop data frames in the length of the} \\ & \text{data block } B. \end{aligned} \quad (10)$$

For queries based on a non-clustered attribute, the hybrid algorithm is similar to the signature method. We assume that control information such as the size of the sparse index tree and the size of the data block is contained at the beginning of each sparse tree to direct the client to the beginning of the next data block. Starting there, the client filters out the requested data frames sequentially. The access time for $Q(a_i)$ where $1 < i \leqslant m$ is

$$\begin{aligned} A^{\text{hyb}}(a_i) &= PROBE^{\text{hyb}} + CYCLE^{\text{hyb}} \\ &= (TREE + SIG + P)\left(\frac{1}{2} + F\right). \end{aligned} \quad (11)$$

In order to skip each of the index tree, we assume that the client needs to retrieve one tree node to get the control information such as the size of the sparse index tree and the size of the data block. Therefore, the tune-in time is

$$T^{\text{hyb}}(a_i) = N + T^{\text{sig}}(a_i), \quad (12)$$

where $T^{\text{sig}}(a_i)$ represents the tune-in time for the corresponding signature scheme used, i.e., multi-level signature. The average access time and tune-in time for multi-attribute hybrid indexing are

$$\begin{aligned} ACCESS^{\text{hyb}} &= p_1A^{\text{hyb}}(a_1) + (1-p_1)A^{\text{hyb}}(a_i), \\ TUNE^{\text{hyb}} &= p_1T^{\text{hyb}}(a_1) + (1-p_1)T^{\text{hyb}}(a_i). \end{aligned}$$

[5] Note that the sparse index tree in the hybrid algorithm is built on the clustered major attribute only.

3.3.2. The hybrid method for multi-attribute queries
The access method for a query $Q\{a'_1 \wedge \cdots \wedge a'_q\}$ is as follows:

If $a'_1 = a_1$

- based on the single index tree access method to retrieve the frame block which contains all the frames satisfying the value for a_1,
- successive signature matches based on $\{a'_2 \wedge \cdots \wedge a'_q\}$ are conducted to filter out the desired frames.

Else based on the signature access method to retrieve the qualified frames based on $\{a'_1 \wedge \cdots \wedge a'_q\}$. The sparse index tree is skipped.

Therefore, the access time for $Q\{a_1 \wedge a_i \wedge a_j\}$, where $1 < i < j \leqslant r$, is

$$A^{\text{hyb}}(a_1 \wedge a_i \wedge a_j) = (TREE + SIG + P)\frac{1+F}{2} + S_1FP.$$

The tune-in time primarily depends on the initial probe of the client to determine the next occurrence of the sparse index, the access time for the index tree part which equals to the number of levels t_1 of the sparse index tree, the tune-in time for the data block B, and the selectivity of the query $Q\{a_1 \wedge a_i \wedge a_j\}$. Therefore, it is bounded by

$$\begin{aligned} T^{\text{hyb}}(a_1 \wedge a_i \wedge a_j) = {} & (1+t_1)N + T(B) \\ & + (1 + S_1S_iS_jF)P, \end{aligned}$$

where $T(B)$ is given in equation (10). For $Q\{a'_1 \vee \cdots \vee a'_q\}$, the sparse index tree part does not help the retrieval of multiple attributes and the access method is similar to the multi-level signature. Thus, the access time for $Q\{a_1 \vee a_i \vee a_j\}$ is

$$\begin{aligned} A^{\text{hyb}}(a_1 \vee a_i \vee a_j) &= PROBE^{\text{hyb}} + CYCLE^{\text{hyb}} \\ &= (TREE + SIG + P)\left(\frac{1}{2} + F\right). \end{aligned}$$

In order to skip each of the index tree, we assume that the client needs to retrieve one tree node to get the control information such as the size of the sparse index tree and the size of the data block. Therefore, the tune-in time for $Q\{a_1 \vee a_i \vee a_j\}$ is

$$T^{\text{hyb}}(a_1 \vee a_i \vee a_j) = N + T^{\text{sig}}(a_1 \vee a_i \vee a_j),$$

where $T^{\text{sig}}(a_1 \vee a_i \vee a_j)$ represents the tune-in time for the corresponding signature scheme used (i.e., multi-level signature).

4. Performance evaluation for multi-attribute data broadcast

In this section, we make analytical comparisons of the multi-attribute data broadcast services based on index tree, signature, and hybrid index methods. Access time, tune-in time, and indexing efficiency for two kinds of queries, single-attribute queries and multi-attribute queries, are compared.

Table 4
Parameters of the cost models.

$F = 1000$–1000000	$P = 1000$	$S = 100$
$p = 64$	$n = 128$	$N = 100$
$k = 4$	$r = 2$	$m = 10$
$p_1 = 0.4$	$p_2 = 0.2$	$p_3 = 0.1$
$\sum_{i=4}^{m} p_i = 0.3$		

For multi-attribute queries, two types of Boolean query expressions, conjunction and disjunction, are investigated.

Table 4 lists the parameter values used in the comparisons. We assume that the broadcast data contains 10 attributes (i.e., $a_1, \ldots, a_{10}$). The access probabilities for a_1, a_2, and a_3 are 0.4, 0.2, and 0.1, respectively. The sum of access probabilities of the remaining attributes is 0.3. Flat broadcast schedule is used for data broadcast. A data frame consists of 1000 packets and a packet contains 64 bits. For index tree, each tree node, which consists of 100 packets, contains 128 search-key plus pointers. Index tree is assumed to be balanced. For signature, the frame group size is 4. To simplify our discussion, we assume that the selectivity of all attributes is the same (i.e., 100).

To show the comparisons clearly, the access time for non-index is used as the baseline[6] and, thus, not shown in the figures. On the other hand, the tune-in time for non-index approach is the upper bound of the indexing methods, so it is shown in the figures for tune-in time comparisons. Based on our evaluation on the impact of the number of indexed attributes to the performance of index tree (not shown due to space limitation), the index tree has the best performance when it indexes only two attributes. Thus, we choose to index two attributes for index tree in the following comparisons.

4.1. Single-attribute queries

In this section, we study the performance of single-attribute queries for multi-attribute data broadcast. As shown in figure 1, the access time of the signature and the hybrid methods have a better performance than the index tree method. Actually, the signature or the hybrid does not increase the access time overhead very much. For a large broadcast cycle (i.e., a cycle with over 5000 frames), the index tree method has very poor access time. On the other hand, figure 2 shows that all of the index methods have better tune-in time performance than the non-index method. This suggests that the multi-attribute indexing methods do improve the tune-in time for single attribute based queries. Among the indexing methods, the index tree has a much worse tune-in time than the other two index methods and the hybrid method gives the best tune-in time.

To correlate the access time and tune-in time performance, figure 3 shows the indexing efficiency of the indexing methods in responding to single attribute queries. We observe that the signature and hybrid index methods have

[6] The access time for non-index is the lower bound of the indexing methods.

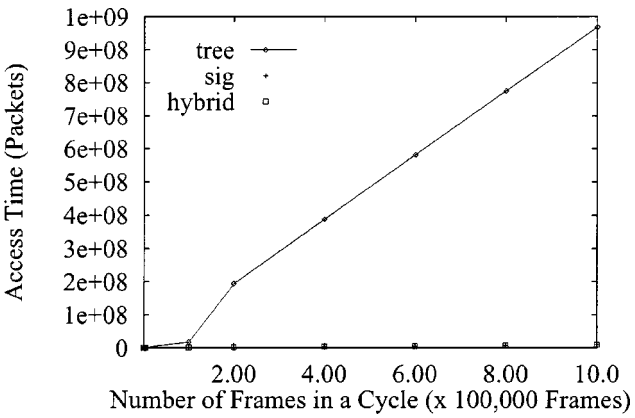

Figure 1. Multi-attribute access time comparisons.

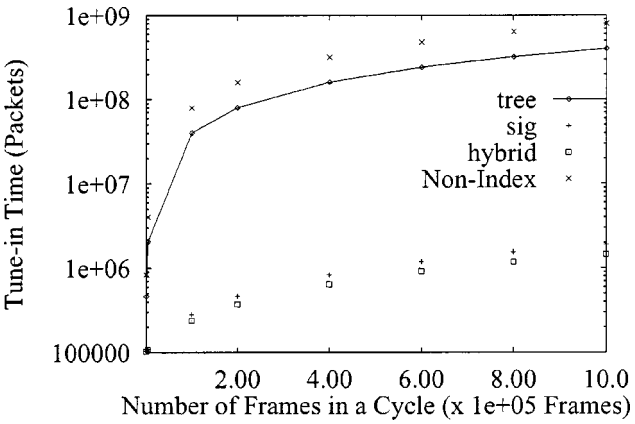

Figure 2. Multi-attribute tune-in time comparison.

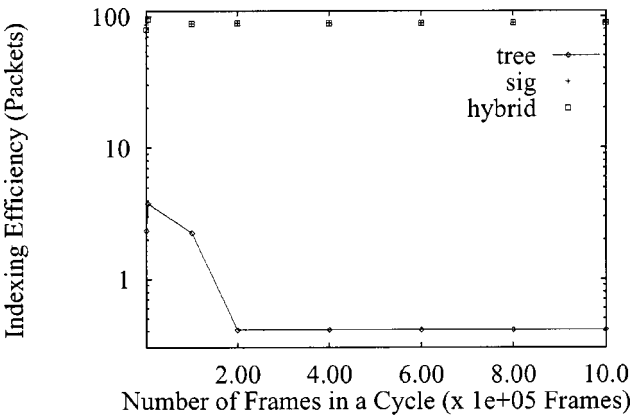

Figure 3. Indexing efficiency as a function of cycle length.

pretty stable index efficiency for various sizes of broadcast cycle. On the other hand, the index efficiency for index tree frist improve, then drop, and finally become stable when the broadcast cycle has more than 200,000 frames. Obviously, the indexing efficiency for the index tree method is the lowest. The other two methods have similar indexing efficiency. Thus, we can conclude that among the three index methods, the hybrid method is the best multi-attribute indexing method for single-attribute queries, followed by the signature method, and the index tree method performs the worst.

4.2. Conjunctive queries

We assume that queries contain a_1, a_2, and a_i where $2 < i \leqslant m$. Figures 4–6 illustrate the access time, the tune-in time, and the indexing efficiency for $Q\{a_1 \wedge a_2 \wedge a_i\}$. The index tree method incurs a large access overhead compared with the non-index method. While the signature and the hybrid methods have similar access time overhead. As shown in figure 4, they perform much better than the index method. In terms of tune-in time, the hybrid index methods is an obvious winner. The signature method is better than the index tree method when the cycle is short. As the size of broadcast cycle increases, the tune-in time for index method outperforms the signature method. This is because the tune-in time consumed for signature filtering is greater than that for traversing index tree. Also, the height of index tree increases very slowly as the cycle increases. As we can see from figure 5, the tune-in time for index tree and hybrid index methods remains nearly constant for different cycle lengths. With the conjunction query, the major attribute based index tree appears to be very efficient and, thus, have a very positive impact on the performance of both the index tree and hybrid methods. Since the signature in the hybrid method provides further filtering capability, the tune-in time for the hybrid method is actually smaller than that for the index tree method and is the best of the three methods. All in all, the three indexing methods give a considerable better tune-in performance than the non-index method. If indexing efficiency is considered (see figure 6), both the signature and the hybrid index methods have the similar efficiency in utilizing index to save tune-in time. The index tree method, however, is much worse in efficiency.

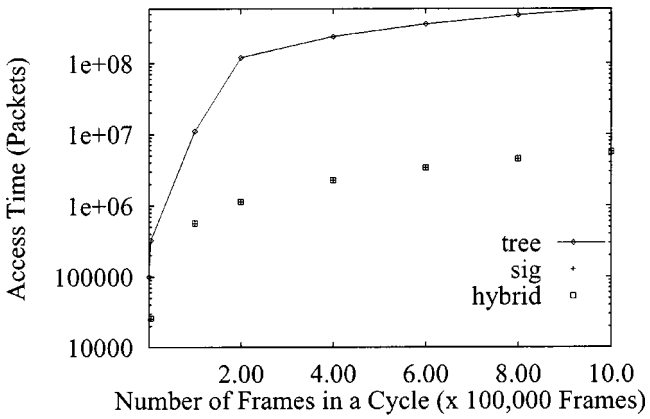

Figure 4. Access time comparison for conjunction query.

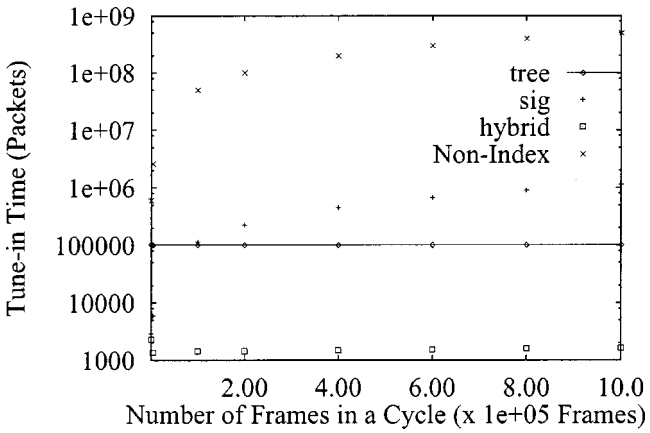

Figure 5. Tune-in time comparison for conjunction query.

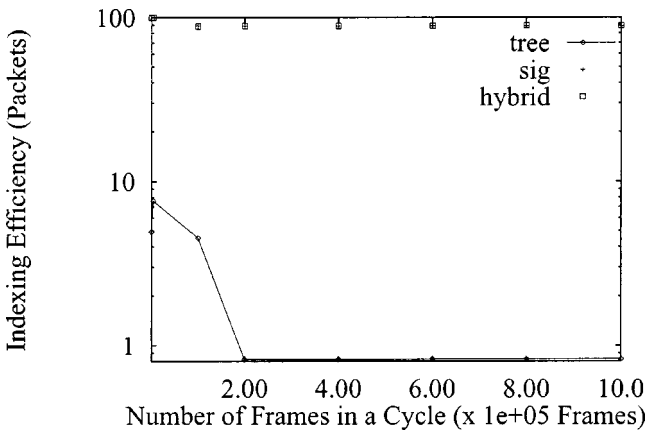

Figure 6. Indexing efficiency comparison for conjunction query.

4.3. Disjunction queries

In figures 7–9, the access time, the tune-in time and the indexing efficiency for $Q\{a_1 \vee a_2 \vee a_i\}$ are shown. Once again, the index tree method has the worst access time among the three index methods and the other two have similar access time close to the non-index method. Figure 8 shows that the index tree has the same tune-in time as the non-index method, while the hybrid and the signature methods give the similar and better performance. This is because we assume that there is no index built on a_i for the index tree method. As a result, the index efficiency for the index tree method is close to 0 (refer to figure 9). On the other hand, the indexing efficiency for the signature and hybrid index methods are similar to their index efficiency for multi-attribute conjunction queries.

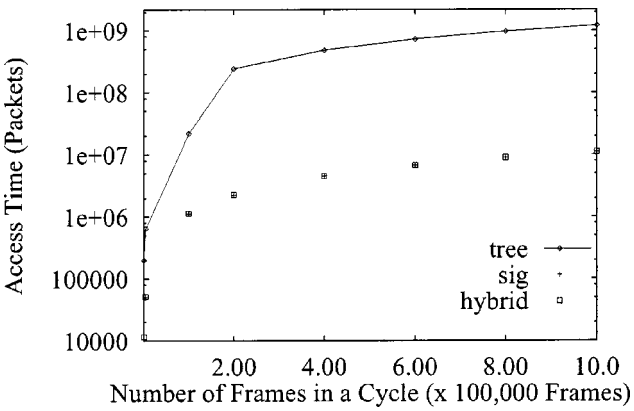

Figure 7. Access time comparison for disjunction query.

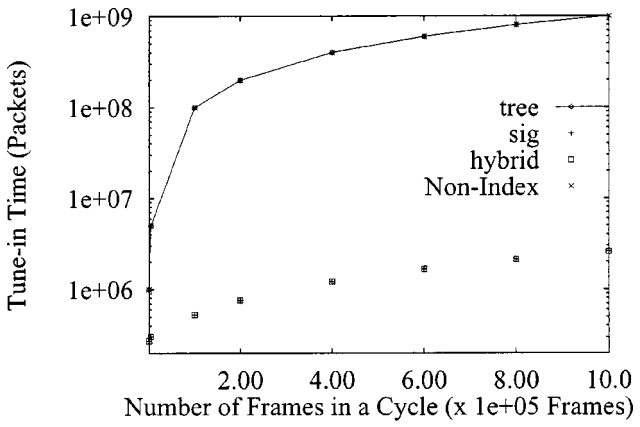

Figure 8. Tune-in time comparison for disjunction query.

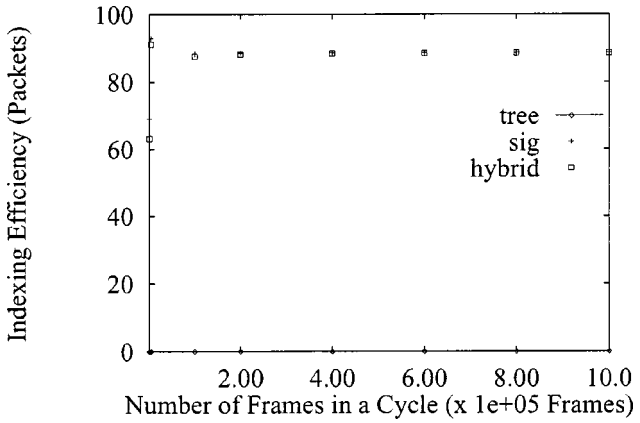

Figure 9. Indexing efficiency for disjunction query.

5. Conclusions and future work

Wireless communications and mobile computing have gradually become a part of our daily life. To support real life applications, multi-attribute data broadcast is an attractive and important service because it can scale up to an arbitrary number of mobile users. In this paper, we address the power conservation problem of multi-attribute data broadcast services by employee the index tree, the signature, and the hybrid index methods. We discuss the data organizations of these three index methods and discussed access methods for processing various queries. To facilitate our evaluation, we assume either that a query is based on one of the attributes which are accessed according to a certain probability or that the query involves multiple attributes conjunctively or disjunctively.

We provide cost models of these three power conserving indexing methods for the estimation of access time and tune-in time. We also compared the performance of the multi-attribute data broadcast services based on these three indexing methods. We found that the distributed index tree method, while performing well for single-attribute queries [5,6,9], results in large access time overhead. This is due to the creation and replication of index trees for all of the indexed attributes. Results of our comparisons show that most of the indexing methods have small access time overhead while improving power consumption by reducing their tune-in time. We conclude from our study that the hybrid and the signature methods are the better methods for multi-attribute indexing than the distributed index method. Between the signature and hybrid index methods, the hybrid index method outperforms the signature method when the number of data frames in a broadcast cycle is large and vice versa when the number of data frames in a broadcast cycle is small.

In this paper, we have studied two multi-attribute queries: conjunction and disjunction. As the next step, we plan to investigate more complicated queries, such as mixed conjunction and disjunction query, join query, and range query. In a mobile computing environment, index information is frequently accessed data. Caching this index information in the clients may reduce both access time and tune-in time considerably. In the future, we will incorporate the index schemes with data caching algorithms to achieve a better system performance.

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] R. Alonso and S. Ganguly, Energy efficient query optimization, Technical report 33-92, Matsushita Information Technology Laboratory (November 1992).

[3] R. Alonso and H. Korth, Database issues in nomadic computing, in: *Proceedings of the ACM SIGMOD Conference on Management of Data*, Washington, DC (May 1993) pp. 388–392.

[4] S. Hameed and N. H. Vaidya, Log-time algorithms for scheduling single and multiple channel data broadcast, in: *The Third International Conference on Mobile Computing and Networking (MobiCom'97)*, Budapest (September 1997).

[5] 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)* (January 1998) pp. 656–659.

[6] Q.L. Hu, W.-C. Lee and D.L. Lee, A hybrid index technique for power efficient data broadcast, Distributed and Parallel Databases (to appear).

[7] T. Imielinski and B.R. Badrinath, Wireless mobile computing: Challenges in data management, Communication of ACM 37(10) (1994).

[8] T. Imielinski and S. Viswanathan, Adaptive wireless information systems, in: *Proceedings of SIGDBS (Special Interest Group in DataBase Systems) Conference*, Tokyo, Japan (October 1994).

[9] T. Imielinski, S. Viswanathan and B.R. Badrinath, Energy efficient indexing on air, in: *Proceedings of the International Conference on SIGMOD* (1994) pp. 25–36.

[10] T. Imielinski, S. Viswanathan and B.R. Badrinath, Power efficient filtering of data on air, in: *Proceedings of the International Conference on Extending Database Technology* (1994) pp. 245–258.

[11] T. Imielinski, S. Viswanathan and B.R. Badrinath, Data on the air – organization and access, IEEE Transactions of Data and Knowledge Engineering 9(3) (May/June 1997) 353–372.

[12] W.-C. Lee, Q.L. Hu and D.L. Lee, A study of channel allocation methods for data dissemination in mobile computing environments, Mobile Networks and Applications (MONET) (to appear).

[13] 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 4(3) (July 1996) 205–227.
[14] W.-C. Lee and D.L. Lee, Signature caching techniques for information filtering in mobile environments, Wireless Networks 5(1) (January 1999) 57–67.
[15] N. Shivakumar and S. Venkatasubramanian, Energy-efficient indexing for information dissemination in wireless systems, Mobile Networks and Applications (MONET) (December 1996).
[16] K. Stathatos, N. Roussopoulos and J.S. Baras, Adaptive data broadcast in hybrid networks, in: *Proceedings of the 23rd VLDB Conference*, Athens, Greece (August 1997) pp. 326–335.
[17] C.J. Su and L. Tassiulas, Broadcast scheduling for information distribution, in: *Proceedings of IEEE INFOCOM'97*, Kobe, Japan (April 1997).
[18] O. Wolfson, P. Sistla, S. Dao, K. Narayanan and R. Raj, View maintenance in mobile computing, in: *SIGMOD RECORD*, Association of Computing Machinery Press (December 1995).
[19] K.-L. Wu, P.S. Yu and M.-S. Chen, Energy-efficient caching for wireless mobile computing, in: *12th International Conference on Data Engineering* (February 26–March 1, 1996) pp. 336–345.

Qinglong Hu received the B.S. and the M.S. degree in computer science from East China Normal University, Shanghai, China, in 1986 and 1989, respectively. After receiving his Ph.D. degree from the Computer Science Department, Hong Kong University of Science and Technology, he is now working as a Postdoc in the Computer Science Department, University of Waterloo, Canada. His research interests include mobile computing and distributed systems.
E-mail: qinglong@styx.uwaterloo.ca

Wang-Chien Lee is a Principal Member of Technical Staff in the Advanced System Laboratory of GTE Laboratories. He received the BS degree in information science from National Chiao Tung University, Hsin-Chu, Taiwan, in 1985; the MS degree in computer science from Indiana University, in 1989, and the PhD degree in computer and information science from Ohio State University in 1996. His current research interests are in the areas of mobile computing, data and metadata management, WWW/XML, Internet information retrieval and integration, and telecommunications management network. Dr. Lee is a member of the IEEE Computer Society, the Association for Computing Machinery, and the Upsilon Pi Epsilon Association.
E-mail: wlee@gte.com

Dik Lun Lee is a Reader of the Computer Science Department at Hong Kong University of Science and Technology. Prior to joining HKUST, he was an Associate Professor of Computer and Information Science at the Ohio State University. In 1992, he was a Distinguished Visiting Scholar at the Online Computer Library Centre (OCLC, Dublin, USA) and a consultant to the Information Dimension Incorporated (IDI, Dublin, USA) on the BasisPlus document management system. Dr. Lee's research interests include document management, access techniques and query processing for text, indexing and search on World Wide Web, object-oriented databases, and mobile computing. He received his PhD in computer science from the University of Toronto. He is currently the Chairman of the ACM Hong Kong Chapter.
E-mail: dlee@cs.ust.hk