Chapter 18

POWER CONSERVING AND ACCESS EFFICIENT INDEXES FOR WIRELESS COMPUTING

Dik Lun Lee & Qinglong Hu
Department of Computer Science
Hong Kong University of Science and Technology, Hong Kong
dlee@cs.ust.hk qinglong@cs.ust.hk

Wang-Chien Lee
GTE Laboratories Incorporated
40 Sylvan Road, Waltham, MA 02451, U.S.A.
wlee@gte.com

Abstract In this paper, we introduce our research on power conserving and access efficient indexing techniques for wireless computing and revisit some of the pioneering work in this area. These indexing methods show that battery power consumption in mobile clients can be significantly reduced with some access time overhead. Among them, *hybrid index* is the best choice for improving power efficiency under both scenarios of single and multiple attributes based queries. Moreover, we introduce two access efficient indexing methods, *cache schedule* and *integrated signature*, for a wireless data dissemination system that incorporates both of *broadcast* and *on-demand* services.

Keywords: power conservation, access efficiency, index, wireless data broadcast, on-demand

1. INTRODUCTION

Wireless communication and mobile computing have received a lot of attention from the computer and communication research community in recent years. Among the various technical problems, *power conservation* and *access efficiency* are two of the most critical issues to be addressed.

Wireless data broadcast is an attractive method for data dissemination, because it allows many users to access the same information at the same time. It also has been used to tackle the issue of power conservation, because mobile clients consume much less power when they are receiving than transmitting data. Many studies addressing various problems in the wireless data broadcast environment have appeared in the literature [8; 1; 9; 16]. A weakness of wireless data broadcast, however, is that a mobile client has to listen to the entire broadcast cycle in order to retrieve all data items of its interests. This not only results in a long access latency, which depends on the length of the broadcast cycle, but also consumes a lot of battery power, since the clients have to remain active for the entire broadcast cycle to receive just a few data items.

Several indexing techniques for wireless data broadcast have been introduced for conserving battery power while maintaining short access latency [11; 13; 2; 15]. Among them, *signature* and *index tree* are two representative methods for indexing broadcast channels [11; 13]. The basic idea behind these techniques is that, by including auxiliary information about the arriving time of data items on the broadcast channels, mobile clients are able to predict the arrival of their data. Thus, they can stay in the power saving mode most of the time and only tune into the broadcast channels when the data items of their interests arrive.

In the literature, three criteria are most frequently used for evaluation of the power conserving indexes:

- *Access Time*: The time elapsed from the moment a mobile client receives a query until all the requested data items are received.
- *Tune-in Time*: The time a mobile client stays active to receive the requested data items.
- *Indexing Efficiency*: The tune-in time saved per unit of access time overhead due to indexing.

While access time measures the efficiency of data access on broadcast channels, tune-in time assesses the power consumption of a mobile computer. The objective of power conserving indexes is to reduce tune-in time, while maintaining an acceptable access time overhead. Thus, indexing efficiency measures how well an indexing method achieves that objective.

When the power conservation is not a concern, the access time alone provides a convenient measurement of the efficiency of a wireless data service. In this paper, we introduce two access efficient indexing techniques, namely, *cache schedule* and *integrated signature*, for a hierarchical data delivery system that incorporates both of wireless broadcast and on-demand services. While this topic is not yet actively pursued by the research community, we believe it's extremely important to a wireless computing system.

The rest of this chapter is organized as follows. In Section 2, we provide some background information regarding to broadcast data broadcast. In Section 3, we discuss various power conserving indexing techniques. We then describe

a hierarchical data delivery system and two access efficient indexing techniques for the system in Section 4. Finally, Section 5 concludes the paper.

2. BACKGROUND INFORMATION

The design of power conserving indexes and access efficient indexes requires undestanding of the mobile computing environments and wireless broadcast related techniques, such as *broadcast clustering* and *scheduling*. In this section, we birelfly introduce wireless services, channel allocation methods, and data organizing techniques for efficient wireless data broadcast.

2.1 WIRELESS SERVICES

A geographical area covered by the wireless system is divided into *cells*. Wireless communication in a cell is supported by a *mobile support station* (MSS). MSSs are stationary computers connected to each other on a fixed network. An MSS is equipped with a wireless interface for communicating with the mobile computers in the cell[1]. In addition, the MSS also provides various data services to the mobile users. Logically, it maintains a database and provides data to its mobile clients. Each data item in the database is uniquely identified by its primary key which is a distinct number assigned by the MSS. The client access pattern is initially unknown to the MSS and is assumed to change with time.

We assume that the *basic fixed channel assignment strategy* [18] is used. That is, a fixed set of channels is permanently assigned to each cell. A service request from a cell can only be served by unoccupied channels in that cell. Otherwise, it is blocked. The communication is asymmetric (i.e., the uplink bandwidth is much less than that of downlink). The wireless channels in a cell provide either one of the following two services:

- **On-demand service:** the channel is bi-directional and asymmetric. The client explicitly makes a pull request to the server via uplink channel. Upon receipt of the request, the server makes connection with the client and returns the required data through the connection. The connection is point-to-point. That is, once it is established between a client and the server, the client occupies the channel until the connection ends.

- **Broadcast service[2]:** the channel is uni-directional. The MSS periodically broadcasts data of popular demands to a large population of clients at the cell site. A complete broadcast of the set of data frames is called a *broadcast cycle*. To retrieve data items, the clients passively monitor the channel and download the data when they arrive.

On-demand and broadcast data services incur different access latency. On-demand service time is determined by the waiting time for the connection and the

transmission time of the data. Since queries are served on a request-by-request basis and an on-demand channels is occupied by a query for the entire duration while it is servered, a large amount of bandwidth is required and it only performs well when the cell is lightly loaded. On the contrary, broadcast service is determined by the volume of data broadcast and the broadcast scheduling. Since data broadcast is shared among all the clients in the cell, wireless bandwidth is more efficiently utilized and the workload of the server is lower compared with on-demand data service. Broadcast service has a good performance when the cell is heavily loaded and when there are data set of common interest to the clients in the cell [4].

2.2 CHANNEL ALLOCATION METHODS

Since broadcast and on-demand modes have different advantages for different system workloads, they can be used together to handle the variation of communication load over time and space due to the mobility of the clients. Generally, there are four channel allocation methods [12; 7; 4]. The *exclusive on-demand* and *exclusive broadcast* methods, respectively, assign all channels in the cell to on-demand or broadcast mode. The *static hybrid* method assigns some of the channels to broadcast mode and some to on-demand mode, and the assignment is fixed in advanced. The *dynamic hybrid* method is an extension of the *static hybrid* method. It dynamically changes the assignment according to system workload and data access patterns. In this paper, we assume that a client gets data through either broadcast channels or on-demand channels but not both. That is, at any time the client is either monitoring the broadcast channels for the desired data to appear or making a pull request to the server for a point-to-point connection.

2.3 EFFICIENT DATA BROADCAST

In addition to indexing techniques, *scheduling* and *clustering* are two important data organizing techniques for improving data access efficiency. Data clustering refers to the consecutive placement of data items with the same value for a specific attribute in a broadcast cycle [10; 11; 7]. 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, a broadcast cycle can only be clustered based on one attribute. Although the other attributes are non-clustered in the cycle, a second attribute can be chosen to cluster the data items within a data cluster of the first attribute. Likewise, a third attribute can be chosen to cluster the data items within a data cluster of the second attribute. 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], 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.

Data scheduling determines the contents of the broadcast cycle and the broadcast frequency of the data frames [16; 1; 17] (i.e., the hot data set for broadcast and the relative broadcast frequency of each frame). A simple broadcast schedule, called *flat broadcast*, broadcasts each data frame once in every cycle [4]. With flat broadcast, the expected access time for a data item on the broadcast channel is the same for all data items (i.e., half a broadcast period). In the real world, client access usually follows a skewed distribution (e.g., Zipf or Gaussian). It is unfair to have the same access time for all data items regardless of their relative importance to the clients. Consequently, another scheduling method, called *broadcast disks*, was proposed [1].

Broadcast disks broadcast important data more often than the other data to reduce the average response time for queries on more important data. The broadcast disk 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 continue to monitor the rest of broadcast cycle.

3. POWER CONSERVING INDEXING

It is expected that the lifetime of batteries will increase only 20% over the next 10 years [14]. Thus, power conservation is a key issue for battery-powered mobile computers. Wireless data broadcast is efficient for disseminating data of common interest to a massive number of mobile users. Furthermore, indexing techniques can be employed to predict the arrival time of a requested data item so that the client can slip into *doze mode* and switch back to *active mode* only when the data of interest arrives, thus substantially reducing battery consumption.

In the following, we present the indexing methods we proposed and the related techniques[3] existing in the literature. Our studies on single and multiple attribute queries are also included [6; 7]. The general access protocol for retrieving indexed data frames involves the following steps:

- Initial probe: The client tunes into the broadcast channel and determines when the next index is broadcast.
- Search: The client accesses the index to find out when to tune into the broadcast channel to get the required frames.
- Retrieve: The client downloads all the information frames of interests.

When no index is used, a broadcast cycle consists of data frames only (called *non-index*). As such, the length of the broadcast cycle and hence the access time are minimum. Since every arriving frame must be retrieved into the client to check against the attribute values specified in the query, the tune-in time is very long and is equal to the access time.

3.1 THE HASHING TECHNIQUE

As mentioned previously, there is a tradeoff between access time and tune-in time of the system which can vary for different applications. Thus, we need flexible data organization methods capable of accommodating different classes of users. Among these indexing techniques, *hashing based schemes* and *flexible indexing method* were discussed in [10].

In hashing based schemes, instead of broadcasting a separate directory with the data, 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 hash function hashes the query key attribute to compute the arrival time of the desired frame; 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 data frames are stored immediately following the frame assigned to them by the hashing function.

Flexible indexing first sorts the data in ascending (or descending) order and then divides the file 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.

On selecting between the hashing scheme and the flexible indexing method, the former should be used when the tune-in time requirements are not rigid and the key size is relatively large compared to the record size. Otherwise, the latter should be used.

3.2 THE INDEX TREE TECHNIQUE

As with a traditional disk-based environment, index-tree methods [11] have been applied to data broadcasts on wireless channels. Instead of storing the locations of disk records, an index tree stores the arrival time of information frames.

Figure 18.1 depicts an example of an index tree for a broadcast cycle which consists of 81 information frames. The lowest level consists of square boxes

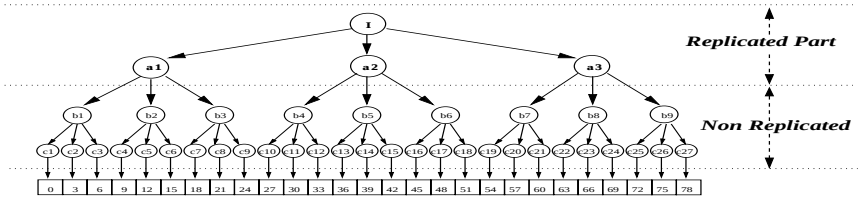

Figure 18.1 A Full Index Tree

which represent a collection of three information frames. The index tree is shown above the information frames. Each index node has three pointers[4].

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 an index replication and interleaving method. The index tree is broadcast every $\frac{1}{d}$ of the file during a broadcast 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 levels of the index tree and each node in that part is replicated a number of times equal to the number of children it has, while the non-replicated part consists of the lower 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 beginning of the root of the next index tree. The first node of each distributed index tree contains a tuple, with the first field containing the primary key of the record that was broadcast last and the second field containing the offset at 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.

3.3 THE SIGNATURE TECHNIQUES

Signature methods have been widely used for information retrieval. A signature of an information frame is basically a bit vector generated by first hashing the values in the information frame into bit strings and then superimposing them together [13]. They are broadcast together with the information frames in

every broadcast cycle. A query signature is generated in a similar way based on the query specified by the user. To answer a query, 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 information frame can be ignored. When the result is the same as the query signature, the information frame still need to be checked against the query. This step is to distinguish a *true match* from a *false drop*.

The signature technique interleaves signatures with their associated information frames. By checking a signature, the mobile computer can decide whether an information frame contains the desired information, if it doesn't, the mobile computer turns into doze mode and wakes up again for the next signature. The primary issue with different signature methods is the size and the number of levels of the signatures.

In [13], 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 information frame. Therefore, the number of signatures is equal to the number of information 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.

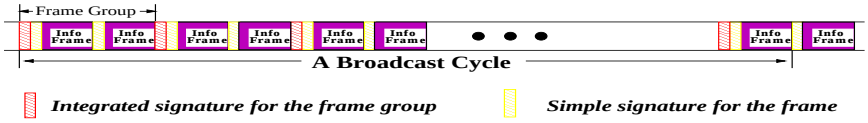

Figure 18.2 The Multi-level Signature Technique

Figure 18.2 illustrates a two-level signature scheme. The white signatures in the figure are integrated signatures. An integrated signature indexes all data frames between itself and the next integrated signature (i.e., two data frames). The black signatures are simple signatures for the corresponding data frames. In the case of non-clustered data frames, the number of data frames indexed by an integrated signature is quite small in order to maintain the filtering capability of the integrated signature. However, if similar data frames are grouped together in each meta segment (as in the case of clustered frames, we can regard the whole file as one meta segment), the integrated signature generated in this file organization has the same effect as reducing the number of bit strings superimposed. Therefore, the number of frames indexed by the integrated signature can be large.

3.4 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 the indexed attribute. Since the clients can quickly search 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.

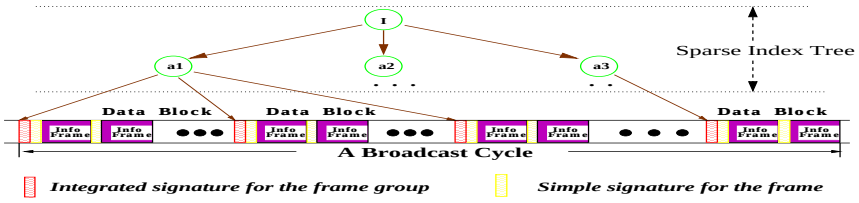

Figure 18.3 The Hybrid of Index Tree and Signature

A new index method, called the *hybrid index*, builds on top of the signatures a *sparse* index tree to provide a global view for the data frames and their corresponding signatures. The index tree is called sparse because only the upper t levels of the index tree (the replicated part in distributed index) are constructed. The *key-search pointer* node in the t-th level points to a *data block* which is a group of consecutive frames following their corresponding signatures. Figure 18.3 illustrates a hybrid index. To retrieve a data frame, 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 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 the 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.

3.5 MULTI-ATTRIBUTE INDEXES

So far the index techniques considered are based on one attribute and can handle single-attribute based queries. In real world applications, data frames usually contain multiple attributes. Thus, multi-attribute based queries can provide more precise information to the users.

Since broadcast channels are linear medium, when compared to single-attribute index and querying, data management and query protocols for multiple attributes are much more complicated. Data clustering and scheduling, which handle single attribute well, are not efficient for multiple attributes. Index information increases the length of the cycle. Index for an attribute influences the performance of queries based on only that attribute and those on other attributes as well.

For multiple attributes, the broadcast cycle is partitioned into meta segments for each attribute with frequently accessed attributes first. The scattering factor of an attribute increases as the importance of the attribute decreases. Organizing data broadcast with the above discussed clustering structure can improve retrieval efficiency.

For multi-attribute data files, a cycle can be organized in broadcast disks based on one of the attributes (i.e., the cycle is not clustered on the other attributes). Due to data frame duplication, queries other than that attribute may take longer to answer. Therefore, broadcast disks are not efficient for data files with multiple attributes. Thus, the flat broadcast scheduling is used for multi-attribute data files.

Index tree, signature, and hybrid are applicable to broadcast of multi-attribute data frames [6]. For the multi-attribute queries, an index tree is built for each index attribute, while multiple attribute values are superimposed to generate signatures.

When two special type of queries, i.e., queries with all conjunction and queries with all disjunction operators, are considered, empirical comparisons show that the index tree method, while performing well for single-attribute queries results in large access time overhead. This is due to the creation and replication of index trees for the indexed attributes. Moreover, the index method has an update constraint, i.e., updates of a data frame are not reflected until the next broadcast cycle. The comparisons revealed that all index methods introduced certain access time overhead while improving power conservation by reducing their tune-in time.

According to experiments the hybrid is the best choice for multi-attribute queries due to its good access time and tune-in time. The signature method has similar performance as the hybrid method except for conjunction queries. The index tree method is poor in access time for any types of multi-attribute queries. It haves similar tune-in time as the hybrid method for the conjunction queries.

4. ACCESS EFFICIENT INDEXING

Wireless computing systems based on hybrid channel allocation methods facilitate a good balance of broadcast and on-demand channels. In addition, it allows the mobile users to use both kinds of channels to maximize their access efficiency. A hierarchical data delivery system has been proposed by the authors [5]. In this section, we introduce access efficient indexing techniques designed for the system and related issues.

4.1 HIERARCHICAL DATA DELIVERY SYSTEM

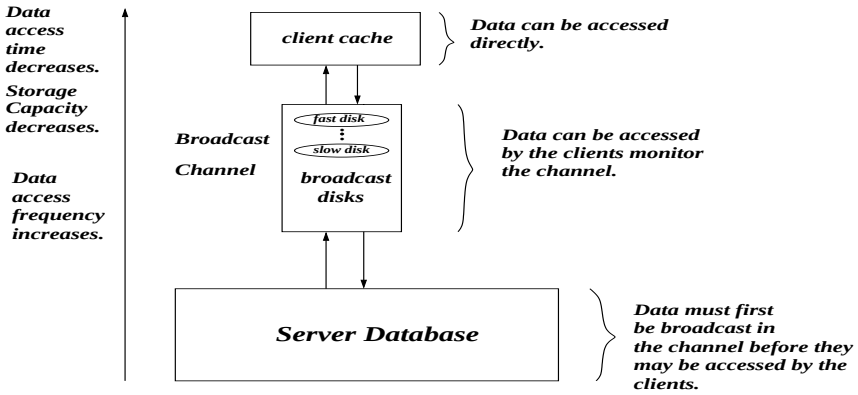

Figure 18.4 Hierarchical Data Distribution.

To minimize client access time for a large population of clients, data caching, data broadcasting and pull-based requests can be used together. Logically we can assume that data are stored in a hierarchical medium (Figure 18.4) such that the most frequently accessed data for a client are cached in the client, the data subset accessed by many clients is temporarily stored in the air (broadcast on the channel), and the rest of the data can be pulled from server via explicit client requests. Data caching and push-based data dissemination can alleviate pull-based request considerably, as most frequently access data can be retrieved either from the client's cache or the broadcast channel.

The hierarchical data delivery system dynamically updates its broadcast schedule based on the client access patterns collected through a bit vector mechanism [3]. Based on the data access patterns, a subset of medium size of frequently accessed data is selected for broadcast on the channels and the broadcast program (i.e., broadcast disks) is designed accordingly.

When a user issues a query to the client, the client first searches the cache. If there is a valid copy in the cache, an answer is replied immediately. Otherwise, the client attempts to obtain the data item from the server site. For a static

hybrid channel allocation, the data access method depends on the availability of indexing methods.

Indexing mechanisms can facilitate the mobile clients of the hierarchical data delivery system to determine when to switch from the push-based broadcast channels to the pull-based on-demand channels. In the following, we discuss two indexing methods, namely, *cache schedule* and integrated signature, and their application in the hierarchical data delivery system. Athough these methods are based on very simple ideas, we expect them to stimulate more studies on the indexing issues of the hierarchical data delivery systems.

4.2 CACHED SCHEDULE AND SIGNATURE

When no index is used, the client doesn't know which data frame is to appear next on the broadcast channels. Thus, pre-scheduling data access for broadcast and on-demand services is impossible. The client first monitors the next *Threshold* number of data frames on the broadcast channels. When the desired data frames are encountered, they are retrieved into the client cache. Otherwise, after monitoring *Threshold* data frames, the client turns to on-demand data service and issues a pull request to the server.

For *cached schedule* method, the complete broadcast schedule is broadcast at the beginning of each cycle. Each active client keeps on monitoring the schedules and retrieves them into its cache. Based on the schedules, the client can make selection between broadcast and on-demand services. That is, if the desired data frames will appear within the next *Threshold* number of data frames on the broadcast channels, the client monitors the broadcast channels. Otherwise, the client issues a pull request to the server immediately. Broadcasting the schedules will take up bandwidth. The biggest drawbacks for this index method are that monitoring broadcast schedule is not energy efficient to the client and that the requirement of all clients remaining active is not flexible.

When an *integrated signature* is broadcast together with the data, the client first monitors the broadcast channels for the data frames and the signatures. If the signatures retrieved indicate that the desired data frame will be broadcast in the cycle and the number of data frames before the desired data to appear in the broadcasting cycle is less than *Threshold*, then the client continues monitoring the broadcast channels and retrieves the items into the cache when they arrive. Otherwise, the client issues a pull request to the server. Although, signature incurs some index overhead, such overhead is low since the signature size is usually very small.

5. CONCLUSION

In a wireless computing environment, power conservation of the mobile clients is a critical issue to be addressed. An efficient power conserving in-

dexing method should introduce low access time overhead, low tune-in time, and produce high indexing efficiency. As another application, indexing makes selection between push-based data dissemination and pulled-based data delivery possible. 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 review various indexing techniques we proposed for wireless computing and revisited some of the related work.

Our studies of the indexing methods takes into consideration the clustering and scheduling factors which may be employed in wireless data broadcast. Combining strengths of the signature and the index tree techniques, a hybrid indexing method is the best choice for power conservation. This method has the advantages of both the index tree method and the signature method and a better performance than the index tree method. A variant of the hybrid indexing method also has been demonstrated to be the best choice for supporting multiple attributes based queries in wireless broadcast environments [6].

In this paper, we also introduce two access efficient indexing techniques, namely, *cache schedule* and *integrated signature*, for a hierarchical data delivery system that incorporates both of wireless broadcast and on-demand services. While this topic is not yet actively pursued by the research community, we believe it's extremely important to a wireless computing system. 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.

Notes

1. In this paper, we use 'mobile computer', 'mobile client', or 'client' interchangeably.

2. An alternative broadcast method is called on-demand broadcast mode. It supports both of broadcast and on-demand services through a broadcast channel and a low bandwidth uplink channel. The users send on-demand requests through the uplink to the server. In response, the server disseminates the requested data items to the clients through the shared broadcast channel.

3. Many of the techniques presented here are actually extended from their original proposals to cooperate the clustering and scheduling techniques.

4. For simplicity, the three pointers of each index node in the lower most index tree level is represented by just one arrow.

References

[1] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik. (1995). Broadcast disks: Data management for asymmetric communications environments. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 199-210, San Jose, California USA.

[2] M.-S. Chen, P. S. Yu, and K.-L. Wu. (1997). Indexed Sequential Data Broadcasting in Wireless Mobile Computing. In *Proceedings of the 17th International Conference on Distributed Computing Systems (ICDCS97)*, Baltimore, Maryland, USA.

[3] Q. L. Hu, D. L. Lee, and W.-C. Lee. (1998a). Dynamic Data Delivery in Wireless Communication Environments. In *Workshop on Mobile Data Access*, pages 213-224, Singapore.

[4] Q. L. Hu, D. L. Lee, and W.-C. Lee. (1998b). Optimal Channel Allocation for Data Dissemination in Mobile Computing Environments. In *Proceedings of the 18th International Conference on Distributed Computing Systems (ICDCS98)*, pages 480-487, Amsterdam, The Netherlands.

[5] Q. L. Hu, D. L. Lee, and W.-C. Lee. (1999a). Data Delivery Techniques in Asymmetric Communication Environments. In *Proceedings of the Fifth Annual International Conference on Mobile Computing and Networking (MobiCom'99)*, pages 163-173, Seattle, Washington, USA.

[6] Q. L. Hu, W.-C. Lee, and D. L. Lee. (2000). Power Conservative Multi-Attribute Queries on Data Broadcast. In *Proceedings of the 16th International Conference on Data Engineering (ICDE'2000)*, pages , San Diego, CA, USA.

[7] Q. L. Hu, W.-C. Lee, and D. L. Lee. (1999b). Indexing Techniques for Wireless Data Broadcast under Clustering and Scheduling. In *Proceedings of the Eighth International Conference on Information and Knowledge Management (CIKM'99)*, pages 351-358, Kansas City, Missouri, USA.

[8] T. Imielinski, and B. R. Badrinath. (1993). Mobile Wireless Computing: Solution and Challenges in Data Management. Technical Report DCS-TR-296, Detartment of Computer Science, Rutgers University.

[9] T. Imielinski, and S. Viswanathan. (1994). Adaptive Wireless Information Systems. In *Proceedings of SIGDBS (Special Interest Group in DataBase Systems) Conference*, Tokyo - Japan.

[10] T. Imielinski, S. Viswanathan, and B. R. Badrinath. (1994). Power Efficiency Filtering of Data on Air. In *Proceedings of the International Conference on Extending Database Technology (ETDB)*, pages 245-258.

[11] T. Imielinski, S. Viswanathan, and B. R. Badrinath. (1997). Data on the Air - Organization and Access. *IEEE Transactions of Data and Knowledge Engineering*.

[12] W.-C. Lee, Q. L. Hu, and D. L. Lee. (1997). Channel Allocation Methods for Data Dissemination in Mobile Computing Environments. In *Proceedings of the 6th IEEE International Symposium on High Performance Distributed Computing (HPDC-6)*, pages 274-281, Portland Oregon, USA.

[13] W.-C. Lee, and D. L. Lee. (1996). 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):205-227.

[14] S. Sheng, A. Chandrasekaran, and R.W. Broderson. (1992). A portable multimedia terminal for personal communications. *IEEE Communications Magazine*, pages 64-75.

[15] N. Shivakumar, and S. Venkatasubramanian. (1996). Engery-efficient indexing for information dissemination in wireless systems. *ACM-Baltzer Journal of Mobile Networks and Nomadic Applications (MONET)*.

[16] K. Stathatos, N. Roussopoulos, and J. S. Baras. (1997). Adaptive Data Broadcast in Hybrid Networks. In *Proceedings of the 23rd VLDB Conference*, pages 326-335, Athens, Greece.

[17] C.J. Su, and L. Tassiulas. (1997). Broadcast Scheduling for Information Distribution. In *Proceedings of IEEE INFOCOM'97*, Kobe, Japan.

[18] S. Tekinay, and B. Jabbari. (1991). Handover and Channel Assignment in Mobile Cellular Networks. *IEEE Communications Magazine*, pages 42-46.