12 Sep 2006

# Pervasive Data Access in Wireless and Mobile Computing Environments

Ken C. K. Lee

Wang-Chien Lee

Sanjay Kumar Madria
*Missouri University of Science and Technology*, madrias@mst.edu



## Recommended Citation

K. C. Lee et al., "Pervasive Data Access in Wireless and Mobile Computing Environments," *Wireless Communications and Mobile Computing*, John Wiley & Sons, Sep 2006.
The definitive version is available at https://doi.org/10.1002/wcm.424

WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
*Wirel. Commun. Mob. Comput.* 2008; **8**:25–44
Published online 12 September 2006 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/wcm.424

# Pervasive data access in wireless and mobile computing environments

Ken C. K. Lee[1], Wang-Chien Lee[1*,†] and Sanjay Madria[2]

[1]*Department of Computer Science and Engineering, 342 IST Building, Pennsylvania State University, University Park, PA 16802, U.S.A.*
[2]*Department of Computer Science, 1870 Miner Circle Drive, 310 Computer Scienece Building, University of Missouri-Rolla, Rolla, MO 65409, U.S.A.*

## Summary

The rapid advance of wireless and portable computing technology has brought a lot of research interests and momentum to the area of mobile computing. One of the research focus is on *pervasive data access*. With wireless connections, users can access information at any place at any time. However, various constraints such as limited client capability, limited bandwidth, weak connectivity, and client mobility impose many challenging technical issues. In the past years, tremendous research efforts have been put forth to address the issues related to pervasive data access. A number of interesting research results were reported in the literature. This survey paper reviews important works in two important dimensions of pervasive data access: *data broadcast* and *client caching*. In addition, data access techniques aiming at various application requirements (such as *time*, *location*, *semantics* and *reliability*) are covered. Copyright © 2006 John Wiley & Sons, Ltd.

KEY WORDS: pervasive data access; wireless and mobile computing; broadcast; caching

## 1. Introduction

The advance of wireless communication and portable computing technologies has sparked a lot of research in the area of pervasive and mobile computing in the last decade. One of the primary goals in these research is to facilitate *pervasive data access*, that is, to allow users to efficiently access data at any place at any time. This unrestricted mode of data access, via mobile computers or devices (e.g., mobile phone, palmtops, laptops, and PDA), fosters a new class of mobile applications such as just-in-time stock trading, news services, and mobile games. A tremendous research effort from academia and industry has been put forth to support these new applications.

While technology has been rapidly advancing, various constraints inherited from limitations of wireless communication and mobile devices remain primary challenges in design and implementation of mobile systems and applications. These constraints include:

*Correspondence to: Wang-Chien Lee, Department of Computer Science and Engineering, 342 IST Building, Pennsylvania State University, University Park, PA 16802, U.S.A.
†E-mail: wlee@cse.psu.edu

Contract/grant sponsor: US National Science Foundation (K.C.K.L.; W.C.L.); contract/grant number: IIS-0328881.
Contract/grant sponsor: EIA-0323630 (S.M.).

- Limited Client Capability. Mobile clients have very limited resources. They are typically powered by short-lived batteries. Efficient energy utilization (by smartly turning on/off hardware parts) can prolong client lifetime.
- Limited Bandwidth. Wireless bandwidth is a scarce resource which needs to be properly allocated in order to maximize various service needs.
- Weak Connectivity. Wireless communication is error-prone. Client disconnections occur frequently, which may be intentional (e.g., to save battery power) or unintentional (e.g., due to signal interference).
- User Mobility. In mobile computing environments, users are free to move. User mobility fosters location-dependent applications which may request data in accordance with the current positions of users.

These constraints make pervasive data access in wireless and mobile computing environments uniquely different from data access in a conventional wired server/client environment. *Wireless data broadcast* has been broadly used to address the issues of limited client resources and wireless bandwidth, while mobile client caching techniques are typically used to handle the problems resulted from user mobility and weak connectivity. Thus, in this paper, research results obtained in these two dimensions of pervasive data access are reviewed. In addition, this survey is extended to cover pervasive data access techniques tailored for various application requirements, in terms of time, location, data semantics, and reliability.

The remainder of this paper is organized as follows. Section 2 outlines a general architecture for pervasive data access, based upon which various techniques are developed. Section 3 and Section 4 survey the wireless data broadcast and mobile client caching techniques. Section 5 reviews a number of pervasive data access techniques tailored for some important application requirements. Finally, Section 6 concludes the paper.

## 2. Pervasive Data Access Architecture

Today, there are many wireless technologies (e.g., Bluetooth, IEEE 802.11, UMTS, Satellite, etc.) that could be integrated to construct a seamless, pervasive data access platform. Although their goals and applications are very different, data access via these wireless technologies can be logically captured by a basic model which consists of an access point (i.e., cellular base station or satellite) and a number of wireless channels. A general architecture is shown in Figure 1. A mobile device can be served by a base station (which supports bi-directional communication) or covered by a satellite (which only supports uni-directional data transmission). All access points and servers are connected to a fast internet backbone. Under this architecture, a base station serves as a gateway between mobile clients and remote servers. On the other hand, a wireless cell can be viewed as a simple client/server environment where a base station functions as a local server disseminating information to clients inside the cell. The servers in this architecture

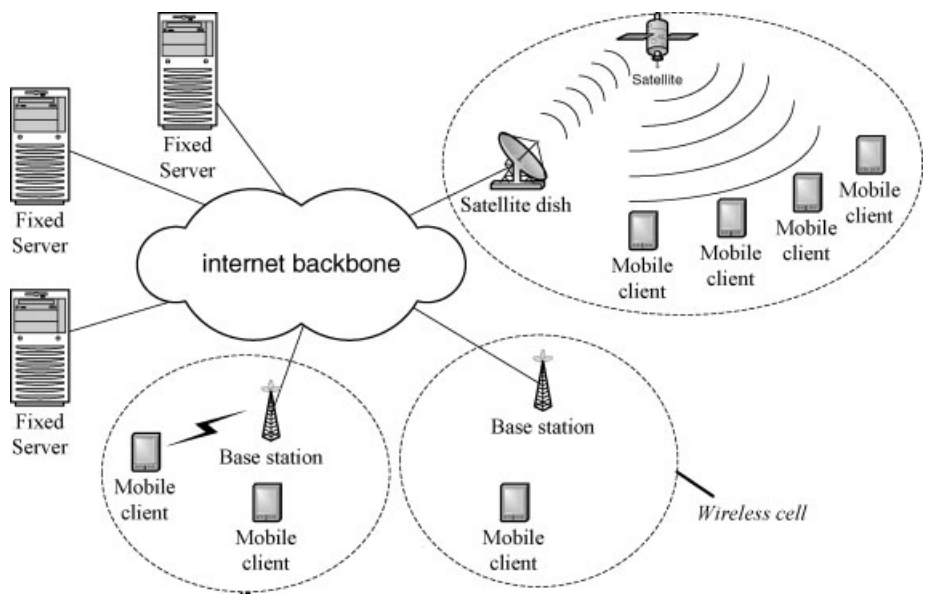


Fig. 1. Pervasive data access architecture.

 

are information providers while mobile clients are customers. Without loss of generality, all updates are performed at the servers.

Typically, the internet backbone is a fast link with data rate in an order of tens of Mbps up to Gbps. Deployed by wireless carriers, modern cellular networks can be based on GPRS [1] and 3G [2] technology. For a single point-to-point access, GPRS can support bandwidth up to 172.2 kbps. The bandwidth of 3G networks can reach 384 kbps for slow moving clients and up to 2 Mbps for stationary clients. Provided by private sectors, Wireless LANs (WLANs) are equipped in some hotels, shopping malls, airports, college campus [3] as Wi-Fi [4] service. 802.11 [5] and its extensions are standard link layer protocols used in WLANs. The bandwidth ranges from 1 Mbps (for 802.11) up to 20 Mbps (for 802.11g) operating at 2.4 GHz. For 802.11b which operates at 5 GHz can provide up to 54 Mbps bandwidth. In general, signal transmitted between a base station (also known as access point) and a client can be as far as 300 ft. If the distance is less than 30 ft, clients operate in low power mode.

Taking a base station as an intermediate host or a proxy [6] forwarding server/client messages, the client and server communication can operate in point-to-point access mode. A client initiates a request to a server and retrieves responses from the server through the base station. Logically, it is identical to conventional client/server interactions in wired networks. However, the wireless link is always a performance bottleneck. To have a better overall system performance, a number of data access techniques are developed. Over wireless medium, information transmitted (i.e., broadcast) can be heard by multiple clients. With this nature, a data item interested by a large portion of client population can be disseminated in a dedicated public channel (called a *broadcast channel*). The clients can tune in and retrieve the desired items by listening to the channel. This technique, called *wireless data broadcast*, significantly saves bandwidth required for delivering the same data item multiple times to individual clients via point-to-point wireless channels. To alleviate the contention of wireless channels, frequently accessed data items can be maintained in the client cache memory. This does not only shorten the access latency but also improve the data availability during the period of disconnection. Due to limited cache size, efficient cache management techniques (including cache replacement, cache coherence and cache prefetching) are required and thus are collectively studied under the theme of mobile client caching.

## 3. Wireless Data Broadcast

Wireless data broadcast (or simply *broadcast*) is particularly efficient for data dissemination in wireless and mobile environments due to its high scalability, efficient client energy conservation, and channel bandwidth utilization. By listening to a broadcast channel, an arbitrary number of clients can be served simultaneously. This not only conserves shared wireless bandwidth but also save client energy (since submitting requests consumes more energy than receiving messages) and alleviate server workload.

For wireless broadcast systems, *access efficiency* and *energy conservation* are two essential performance criteria. Access efficiency concerns how fast a client request is answered, while energy conservation concerns how much battery power a mobile client consumes to access the requested data items. Energy conservation is particularly important due to constrained battery power available to a mobile client. In the literature, two performance metrics, namely *access time* and *tuning time* are typically used to measure the access efficiency and energy conservation of a wireless data broadcast system, respectively.

- Access time. It is the time elapsed from the moment a request is issued by a client to the moment the requested data is returned.
- Tuning time. It is the duration of time a client stays in active mode to collect requested data items.

Broadcast of unwanted data items not only consumes the scarce wireless bandwidth but also increases expected client access time. Thus, *broadcast scheduling*, to arrange the order of data items, and the frequency of these data items in a broadcast channel, to improve the client access time, is a critical research issue. In general, broadcast schedules can be categorized as *push-based broadcast*, *pull-based (on-demand) broadcast*, and *hybrid broadcast*. In push-based broadcast, a server schedules public information (such as weather and traffic information) over a broadcast channel. This approach assumes uni-directional communication (i.e., server to mobile clients only). The server compiles a broadcast program based on the prior knowledge of access profiles or access statistics. In on-demand broadcast, clients send requests for data items to the server via uplink channels. The server then responds the clients by disseminating requested data via a

shared broadcast channel. Different from conventional point-to-point communication, a broadcast data item may satisfy multiple client requests at the same time. Both push and on-demand broadcast can be combined into a hybrid broadcast approach.

A client stays connected to monitor an entire broadcast cycle for a requested data item, resulting in overconsumption of client's precious energy. With Smart power management facility, clients are able to efficiently switch between active and doze modes according to a time schedule and synchronize with a broadcast channel. Thus, *air indexing* techniques have been proposed to let a client effectively turn into sleep mode (to conserve energy) when irrelevant data items are broadcast on air. The basic idea is to incorporate auxiliary information about content and arrival time of data items in the broadcast program. The inclusion of air index, however, lengthens the broadcast cycle and degrades the access efficiency. Hence, a trade-off between access efficiency and tuning time is established.

For the rest of this section, a number of broadcast scheduling techniques via push-based, on-demand, and hybrid broadcast models and air indexing techniques are reviewed.

## 3.1. Broadcast Scheduling

### *3.1.1. Push-based broadcast*

The server actively disseminates data items in a broadcast channel. The broadcast schedule determines the order and the frequencies of the data items to be broadcast. There are four typical methods for push-based broadcast, namely *flat broadcast*, *probabilistic-based broadcast*, *broadcast disks*, and *optimal scheduling*.

- Flat Broadcast. It is the simplest form of broadcast. All data items are broadcast only once in a cycle. The access time of a data item is expected to be the same, that is, a half of the broadcast cycle. If the access distribution is skewed, poor access performance will be resulted.
- Probabilistic-Based Broadcast [7]. It selects data item $i$ from a set of $N$ data items to broadcast with a broadcast frequency, $f_i$. Supposed $q_i$ is the access probability for data item $i$. The best setting of $f_i$ is determined by the rate of square root of $q_i$ to the sum of square roots of access probabilities for all items, that is, $f_i = \sqrt{q_i} \big/ \sum_{j=1}^{N} \sqrt{q_j}$. This approach is deficient since an arbitrarily long access time for a data item may be resulted. It shows an inferior performance to other broadcast algorithms for skewed access distributions [7].
- Broadcast Disk [8]. In broadcast disk schedule, data items of same range of access probability are grouped into *logical disks*. Each disk is assigned a broadcast frequency. The higher access probability disk is assigned the higher broadcast frequency. Such design resembles real physical disks spinning at different speed, with faster disks placing more instances of a data item on the broadcast. Then a broadcast is decomposed into several minor cycles. The length of a minor cycle is long enough to contain one data item from each disk. The total number of the minor cycles is the least common multiple (LCM) of the relative frequencies (spinning speed) of the disks. Figure 2 illustrates the construction of a broadcast program based on 7 data items divided into three groups of access probabilities (i.e., for three separate logical disks). These three disks are interleaved in a single broadcast cycle. The first disk (the fastest one) rotates at a speed twice faster than second one and four times faster than the third disk (the slowest one). The final broadcast consists of four minor cycles. It can be observed that broadcast disk can be used to construct a fine-grained memory hierarchy such that data items of higher probabilities are broadcast more frequently by varying the number of disks, the size, the relative spinning speed, and the number of assigned data items of each disk.
- Optimal Push Scheduling. Optimal broadcast schedules have been studied in References [7,9–11]. A square-root rule discovered in Reference [9] for minimizing access latency states that the minimum overall expected access latency is achieved when two conditions are met. First, instances of each data items are equally spaced on the broadcast channel. Second, the spacing, $s_i$, between two consecutive instances of each item, $i$, is proportional to the square-root of its item length $l_i$ on the broadcast channel and

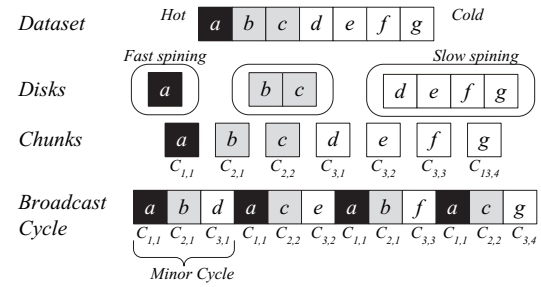


Fig. 2. Broadcast disk example with 7 data items, three disks.

inversely proportional to the square-root of its access probability $q_i$, that is, $s_i \propto \sqrt{l_i/q_i}$ or expressed as $s_i^2 q_i / l_i = K$, where $K$ is a constant for all items $i$.

Since these two conditions are not always simultaneously achievable, an online scheduling algorithm can only provide an approximation of the theoretical results. An efficient heuristic scheme was introduced in Reference [11]. This scheme maintains two variables, $b_i$ and $c_i$ for each data item $i$. $b_i$ is the earliest broadcast time of the next instance of data item $i$ and $c_i$ is maintained equal to $b_i + s_i$. $c_i$ could be interpreted as the 'suggested worst-case completion time' for the next transmission of data item $i$. The algorithm with complexity $O(\log N)$ for $N$ data items operates iteratively. At the very beginning, every data item $i$ is initialized with $s_i$ and a global variable $t$ is set to the current time. Then the algorithm examines all the data items and selects a data item $j$ with minimum $c_j$ for broadcast. Its $b_j$ and $c_j$ are updated to old value $c_j$ and $b_j + s_j$, respectively, while $t$ is revised to $t + l_j$.

A low-overhead bucket-based scheduling algorithm based on square-root rule was also studied in Reference [9]. In this strategy, the data set is partitioned into several buckets which are kept as separate cyclical queues. The algorithm chooses to broadcast the first item in the bucket for which the expression $(T - R(I_m))^2 q_m / l_m$ evaluates to the largest value. In the expression, $T$ is the current time, $R(i)$ is the time at which an instance of item $i$ was most recently transmitted, $I_m$ is the first item in bucket $m$, and $q_m$ and $l_m$ are average values of $q_i$ and $l_i$ for the item in the bucket $m$.

The bucket-based scheduling algorithm is similar to the broadcast disk. They differ in the following aspects. Broadcast disk generates periodic broadcast while bucket-based approach delivers data item online. Bucket-based approach fills up data item instances on the scheduling decision whereas broadcast disk may create holes in its broadcast program. Last, the broadcast frequency for each data item is obtained analytically to achieve the optimal overall system performance in the bucket-based algorithm. However, no study has been carried out to compare their performance.

### 3.1.2. On-demand broadcast

Push-based broadcasts are designed to serve the majority of the clients, but not tailored to the individual client needs. In addition, they tend to limit the broadcast data items to a pool and react slowly to changing client access patterns. To alleviate these problems, several research studies propose to use on-demand broadcast [10,12–14]. An on-demand broadcast system supports two types of channels, a high-bandwidth broadcast channel plus a low-bandwidth uplink channel. When a client needs data items, it sends to the server a request for them through a uplink channel. All client requests are queued at the server. The server repeatedly chooses requested items, delivers them over the broadcast channel, and clear the associated requests in the queue. The client monitors the broadcast channel and retrieves her requested items. In the following, on-demand broadcast scheduling techniques for fixed-size items and variable-size items, and energy-efficient on-demand scheduling are described.

- Fixed-Size Item On-demand Broadcast. Scheduling fixed-size items in a broadcast is considered in most early studies. The average access time performance was used as the main optimization objective. In Reference [15] (also described in Reference [7]), three scheduling algorithms were proposed and compared with the FCFS algorithm, a naive approach.
  - — First-Come-First-Served (FCFS). Data items are broadcast according to the arrival order of their corresponding requests. This simple scheme has a poor access performance for skewed access distribution.
  - — Most Requests First (MRF). Data items with the largest number of outstanding requests are broadcast first.
  - — MRF Low (MRFL). This scheme is essentially the same as MRF but it breaks the tie in favor of the item with the lowest request probability.
  - — Longest Wait First (LWF). Data items with the largest total waiting time, that is, the sum of the time that all outstanding requests for the item has been waiting, is chosen for broadcast first.

  Numerical results presented in Reference [15] provide the following observation. When the system load is light, the average access time is insensitive to the scheduling algorithm used. This is expected because few scheduling decisions are required in this case. As the load increases, MRF yields the best access time performance if request probabilities on the items are equal. When request probabilities follow the *Zipf* distribution [6], LWF has the best performance while MRFL is close to LWF. However,

LWF is not a practical algorithm because at each scheduling decision, it needs to recalculate the total accumulated waiting time for every data item against all pending requests to decide the next broadcast candidate.

Thus, MRFL was suggested as a low-overhead replacement of LWF in Reference [22]. It was, however, observed in Reference [13] that both MRFL and MRF have a poor performance for a large data set. This is because the opportunity for tie-breaking diminishes and thus MRFL degenerates to MRF. Consequently a low-overhead and scalable approach called $RxW$ was proposed in Reference [11]. The $RxW$ algorithm schedules a data item with the maximal $R \times W$ value for the next broadcast, where $R$ is the number of outstanding requests for that item and $W$ is the amount of time that the oldest of those requests has been waiting. Thus, $RxW$ broadcasts an item either because it is very popular or because it has a pending request waiting for a long time. The method could be implemented efficiently by maintaining the outstanding requests in two sorted queues, one ordered by $R$ values and the other ordered by $W$ values. Avoiding exhaustive search of the request queues, a pruning technique was proposed to find the maximal $R \times W$ value. Simulation results show that the performance of the $RxW$ is close to LWF, meaning that it is a good alternative for LWF when scheduling complexity is a major concern.

To further reduce scheduling overheads, a parameterized algorithm was developed to extend $RxW$. The parameterized $RxW$ algorithm selects the first item it encounters in the searching process whose $R \times W$ value is greater than or equal to $\alpha \times threshold$ where $\alpha$ is a system parameter and $threshold$ is the running average of the $R \times W$ values of the requests that have been served. Varying $\alpha$ can adjust the performance trade-off between access time and scheduling overhead. For example, in the extreme case where $\alpha = 0$, this scheme selects the top item either in the $R$ list or in the $W$ list; this has the least scheduling complexity, while its access time performance may not be very good. With larger $\alpha$ values, the access performance can be improved, but the scheduling overhead is increased as well.

- Variable-Size Item On-demand Broadcast. On-demand broadcast for variable data item sizes was studied in Reference [12]. To evaluate the performance for items of different sizes, a new performance metric called *stretch* was used. Stretch is a ratio of the access time of a request to its service time, where the service time is the time needed to complete the request.

  Compared with the access time, the stretch performance is believed to be a more reasonable metric for data items of variable sizes since it takes the size (i.e., the service time) of a requested data item into consideration. Based on the stretch metric, four different algorithms have been investigated [12]. All of the four algorithms considered are pre-emptive in the sense that the scheduling decision is re-evaluated after broadcasting any bucket of a data item (assuming a data item covers multiple buckets).

  — Pre-Emptive Longest Wait First (PLWF). This is the pre-emptive version of the LWF algorithm. The LWF criterion is applied to select the subsequent buckets of data items to broadcast.
  — Shortest Remaining Time First (SRTF). The data item with the shortest remaining time is selected to broadcast first.
  — Longest Total Stretch First (LTSF). The data item which has the largest total current stretch is chosen to broadcast first. Here, the current stretch of a pending request is the ratio of the time the request has been queued in the system thus far to its service time.
  — MAX algorithm (MAX). MAX was proposed based on an off-line algorithm [12]. MAX assigns a deadline to each request when it arrives at the server. The data item with the earliest deadline is scheduled for the next broadcast. In computing the deadline for a request, the following formula is used:

  $$deadline = arrivaltime + servicetime \times S_{\mathrm{MAX}}$$

  where $S_{\mathrm{MAX}}$ is the maximum stretch value of the individual requests for the last satisfied requests in a history window. To reduce the computational overhead, once a deadline is set for a request, this value does not change even if $S_{\mathrm{MAX}}$ is updated before the request is served.

  The trace-based performance study carried out in Reference [12] indicates that none of these schemes is superior to the others for all cases. Their performance really depends on the system settings. Overall, the $MAX$ scheme, with a simple implementation, performs quite well in both the worst and average cases in terms of access time and stretch measures.

- Energy-Efficient On-Demand Broadcast. The study in Reference [17] takes energy saving issue into consideration in on-demand broadcasts. The proposed algorithms broadcast the requested data items in batches, using an existing indexing techniques [18] (see Subsection 3.2) to index the data items in the current broadcast cycle. This way, a mobile client may tune into a small portion of the broadcast instead of monitoring the broadcast channel until the desired data arrives. Thus, the proposed method is energy efficient. The data scheduling is based on a priority formula:

$$Priority = IF^{ASP} \times PF$$

where Ignore Factor (*IF*) denotes the number of times that the particular item has not been included in a broadcast cycle, Popularity Factor (*PF*) is the number of requests for this item, and Adaptive Scaling Factor (*ASP*) is a factor that weighs the relative significance of $IF$ and $PF$. Two sets of broadcast protocols, namely *Constant Broadcast Size* (*CBS*) and *Variable Broadcast Size* (*VBS*), were investigated in Reference [17]. The CBS strategy broadcasts data items in decreasing order of the priority values until the fixed broadcast size is exhausted. The VBS strategy broadcasts all data items with positive priority values. Simulation results show that the VBS protocol outperforms the CBS protocol at light system loads, while the CBS protocol predominates the VBS protocol at heavy system loads.

### 3.1.3. Hybrid broadcast

Although on-demand data broadcast can schedule data items depending on the needs of users, it comes with two disadvantages: (i) more uplink bandwidth is allocated and more client energy is consumed to submit client requests; (ii) when an uplink channel is congested, the access latency will become extremely high. Hybrid broadcast is a promising approach because it effectively combines both push-based and on-demand techniques. In the design of a hybrid broadcast system, three issues need to be considered: (1) bandwidth allocation between push and on-demand deliveries, (2) assignment of data items to either push-based and on-demand broadcast channels; and (3) client access methods to collect required items from appropriate channels. There are different proposals for hybrid broadcast in the literature to address these issues. In the following, techniques for balancing push and pull and adaptive hybrid broadcast are described.

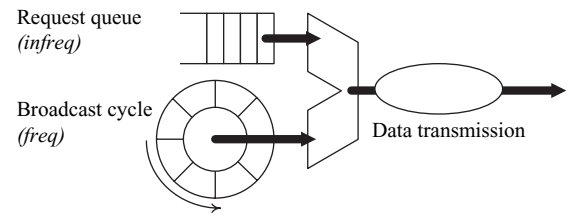


Fig. 3. Hybrid broadcast system model.

- Balancing Push and Pull. The hybrid model (shown in Figure 3) was firstly investigated in References [15,19]. In this model, items are put in two distinct sets: frequently requested (*freq*) or infrequently requested (*infreq*) sets according to items' access frequency. It is assumed that clients know *freq* and *infreq*. The model puts *freq* in a broadcast cycle and delivers *infreq* in a request queue on demand. In the downlink scheduling, the server makes $r$ consecutive transmission of *freq* items followed by the transmission of one *infreq* items. Analytical results for the average access time were derived in Reference [19].

  Meanwhile, the push-based broadcast disk model was extended in Reference [20] to integrate with a pull-based approach. The proposed hybrid solution, named Interleaved Push and Pull (IPP), consists of an uplink for clients to send pull requests to the server for the data items absent on the push-based broadcast. The server interleaves the broadcast with the responses to pull requests on the channel. To improve the scalability of IPP, three different techniques were proposed: (i) adjusting the bandwidth assignment to push and pull in term of relative frequency of items to broadcast, (ii) setting a client patience (time the client willing to wait a data item) [21,22] such that a client pulls the item only if the waiting time for a item exceeds its patience on broadcast; (iii) selectively replacing pushed items from the slowest part of the broadcast schedule with pulled items (i.e., in effect reallocating more bandwidth from pushes to pulls).
- Adaptive Hybrid Broadcast. Adaptive broadcast strategies were studied for dynamic systems [23,24]. These studies are based on the hybrid model where the most frequently accessed items are delivered to clients based on a flat broadcast, while the least frequently accessed items are provided point-to-point on a separate channel. In Reference [24], a technique that continuously adjusts the broadcast content to match the hot spot of a data set was proposed. To do this, each item is associated with a temperature, a metric corresponding to its request rate. Thus, each item can be in either *vapor*, *liquid* and *frigid* states.

Vapor data items are those frequently requested and currently broadcast; liquid data items are those having recently a moderate number of requests which is still not large enough for immediate broadcast; frigid data items refer to the cold items. The access frequency and hence the state of a data item can be dynamically estimated from the number of on-demand requests received through the uplink channel. For example, liquid data items can be heated to vaporize if more requests are outstanding. Simulation result show that this technique adapts pretty well to rapidly changing workloads.

Another adaptive broadcast scheme was discussed in Reference [23], which assumes fixed channel allocation for data broadcast and point-to-point communication. The idea for adaptive broadcast is to maximize (but not to overload) the use of available point-to-point channels so that a better overall system performance can be achieved.

## 3.2. Air Indexing

Energy conservation is a key issue for battery-powered mobile devices. Air indexing techniques are employed to indicate the arrival times of data items. Instead of full scanning a broadcast channel for requested data items, the broadcast access with an air index involves the following steps: (i) *initial probe*—tuning into the broadcast channel to determine when an index is broadcast; (ii) *index lookup*—accessing the index to determine the time when a required data item is on air; (iii) *download*—retrieving required data item at its indicated broadcast time. A client can switch into *doze mode* and resume to *active mode* only when the interested index or data items are expected to arrive, thus substantially reducing battery power consumption. In the following, three basic classes of indexing techniques namely *hashing*, *index tree* and *signatures* are described:

- Hashing Technique and Flexible Indexing. Both hashing-based schemes and flexible indexing method were proposed in Reference [25]. In hash-based schemes, each frame carries the control together with the data inside the frame. The control information consists of a hash function and a shift function that guide a search to the frame containing the desired data in order to improve the tuning time. The hash function hashes a key attribute to the address (i.e., delivery time) of the frame holding the desired data. In the case of collision, the shift function is used to compute the address of the overflow area which consists of a sequential set of frames starting at a position behind the frame address generated by the hash function.

  Flexible indexing first sorts the data items in ascending (or descending) order and then divides them into $p$ segments. The first frame in each of the data segments contains a control index, which is a binary index mapping a given key value to the frame containing that key. In this way, the tuning time can be reduced. The parameter $p$ makes the indexing method flexible since either a very good tuning time or a very good access time can be obtained by selecting a $p$ value.

  Hash-based scheme should be used when the tuning time requirement is not tight and the key attribute size is relatively large compared with the data item size. Otherwise, flexible indexing should be used.
- Index Tree Technique. An index tree [18] stores the key attribute values and the arrival times of the data frames in each node disseminated in a broadcast channel. The index tree technique is very efficient for a clustered broadcast cycle and it provides a more accurate and complete global view of the data frames. An example of an index tree for a broadcast cycle which consists of 81 data items is shown in Figure 4. The lowest level consists of 27 rectangular boxes each of which represents a data frame, that is, a collection of three data items. Each index node has three pointers.

  To reduce the tuning time while maintaining a good access time for clients, a part of index tree can be replicated and interleaved with the data frames. Instead of replicating the entire index tree $m$ times (meaning the index tree is broadcast every $1/m$ of the broadcast cycle), each broadcast only consists of the replicated part of the index (the upper levels) and the non-replicated part (the lower levels) that indexes the data frames immediately following it. As such, each node in the non-replicated part appears only once in a broadcast cycle. Since the lower levels of an index tree take up much more bandwidth than the upper part, the index overheads can be greatly reduced if the lower levels of the index tree are not replicated. This scheme is termed *distributed index* [18]. In this way, tuning time can be significantly improved without causing much deterioration in access time.

  To support distributed indexing, every frame maintains an *offset* to the beginning of the index root to be broadcast. Each node of an distributed index tree contains a tuple, with the first field containing the last primary key value of the data frame that is previously broadcast and the second field containing

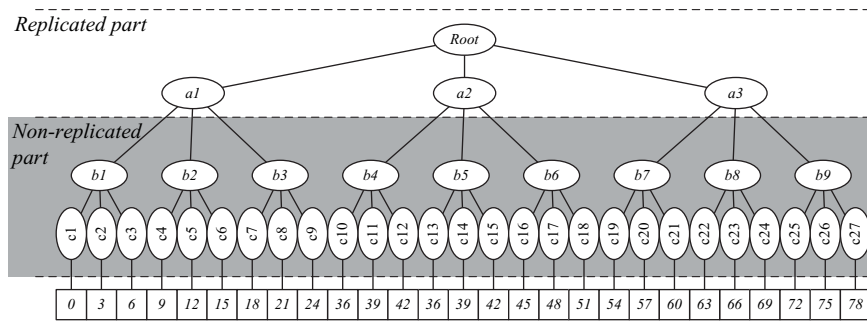


Fig. 4. Full index tree.

the offset to the beginning of the next broadcast cycle. This is to guide the clients that have missed the required data in the current cycle to tune to the next broadcast cycle. There is a *control index* at the beginning of every replicated index to refer clients 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.

- Signature Technique. The signature technique has been widely used for information retrieval. The signature method is not affected much by the clustering factor and thus is particularly good for multi-attribute retrieval. The signature of a data frame is generated by first hashing the values in the data frame into bit strings and then superimposing (using bitwise OR operation) them into a bit vector [26]. Signatures are delivered in the broadcast channel by interleaving them with the data frames. A query signature is generated in the same fashion based on the queried values given by the user. To answer a query, a mobile client can simply retrieve signatures from the broadcast channel and then match the signatures with the query signature by performing a bitwise AND operation. If no match is determined, the corresponding data frame determined not to contain requested items can be safely ignored. The client turns into doze mode and wakes up again for the next signature. Otherwise, the data frame is further checked against the query. The primary issue with different signature methods is the size and the number of levels of the signatures to be used.

  In Reference [26], three signature schemes, namely *simple signature*, *integrated signature*, and *multi-level signature* were proposed. Their cost models for access time and tuning time were given. For simple signatures, a signature frame is broadcast before its corresponding data frame. Therefore, the number of signatures is equal to the number of data frames in a broadcast cycle. An integrated signature is constructed for a group of consecutive data 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 5 illustrates a two-level signature scheme. An integrated signature indexes all data frames between itself and the next integrated signature (i.e., two data frames). The signatures (shown in gray blocks) are simple signatures for the corresponding data frames. In case of non-clustered data frames, the number of data frames indexed by an integrated signature (in a black block) is usually small in order to maintain the filtering capability of the integrated signatures. On the other hand, if data frames with similar contents are grouped together, the number of frames indexed by an integrated signature can be large.

  Among these basic schemes, a comprehensive performance study of hash, index, and signature-based index schemes was reported in Reference [27].

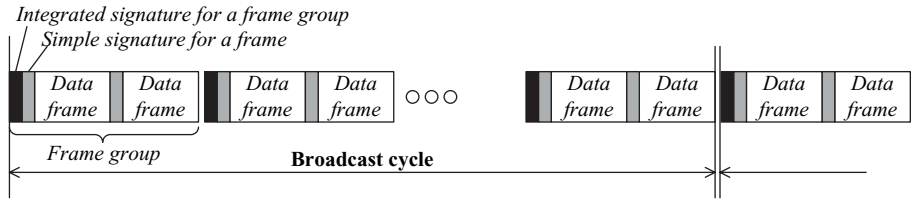


Fig. 5. Multi-level signature technique.

In general, signature indexing achieves better access time than most of other schemes. However, the tuning time of signature schemes is comparatively larger. Hashing usually achieve better tuning time. In case of low data availability (caused by disconnection) and key size is relatively small compared with the record size, distributed indexing achieves both good access time and tuning time.

Derived from these basic schemes, enhanced indexing techniques namely hybrid index, unbalanced index tree, and multi-attribute air indexing are discussed in the following.

- Hybrid Index Approach. Both the signature and index tree techniques have some pros and cons. Hybrid index proposed in Reference [28] builds index information on top of the signatures. A sparse index tree provides 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 the distributed indexing) are constructed. A key-search pointer node in the $t$-th level points to a frame group, a group of consecutive frames following their corresponding signatures. Since the size of the upper $t$ levels of an index tree is usually small, the overheads for such additional indexes are very small. Figure 6 illustrates a hybrid index. To retrieve a data frame, a mobile client first searches the sparse index tree to obtain the approximate location information about the desired data frame group and then tunes into the broadcast to find out the desired frame.

  Since the hybrid index technique is built on top of the signature method, it retains all of the advantages of a signature method. At the same time, the global information provided by the sparse index tree considerably improves the tuning time.

- Unbalanced Index Tree Technique. To achieve better performance with skewed queries, the unbalanced index tree technique was investigated in References [29,30]. Unbalanced indexing minimizes the average index search cost by reducing the number of index traversals for hot data items at the expense of spending more on cold ones.

  For fixed index fanouts, a *Huffman-based* algorithm can be used to construct an optimal unbalanced index tree. Given $N$ data items and fanout $f$ of the index tree, the Huffman-based algorithm first creates a forest of $N$ subtrees, each of which is a single node labeled with the corresponding access frequency. Then, $f$ subtrees labeled with the smallest frequency are attached to a new node. The resulting subtree inherits the sum of all the labeled frequencies from all its $f$ child subtrees. This grouping procedure is repeated until only one single tree remains. Figure 7(a) demonstrates an index tree with a fixed fanout of three. In the figure, each data item $i$ is given in the form of $(i, q_i)$ where $q_i$ is the access probability for item $i$.

  Provided the data access patterns, an optimal unbalanced index tree with a fixed fanout can be constructed. However, its performance may not be optimal. Thus, Reference [29] discussed a more sophisiticated approach with variable fanouts. In this case, the problem of optimally constructing an index tree is NP-hard [29]. In Reference [29], a greedy algorithm Variant Fanout (VF) was proposed. Basically, the VF algorithm builds the index tree in a top-down manner. VF starts by attaching all data items to the root node. Then, it groups the nodes with small access probabilities and moves them to one level lower so as to minimize the average index traversal cost. Figure 7(b) shows an index tree built using the VF algorithm, where the access probability for each data item is the same as in the

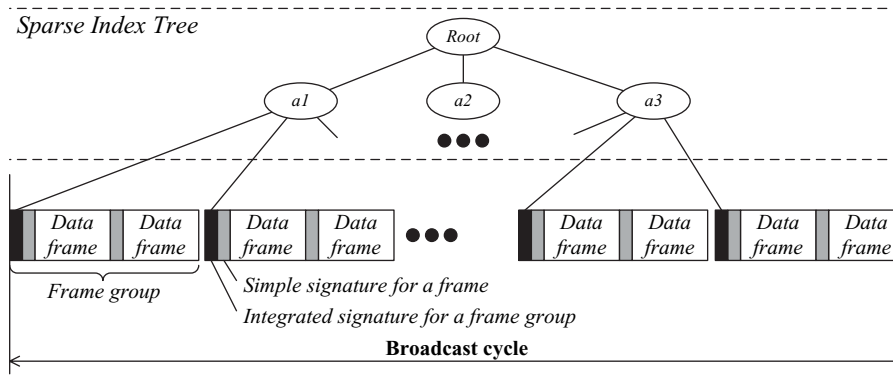


Fig. 6. Hybrid index technique.

 

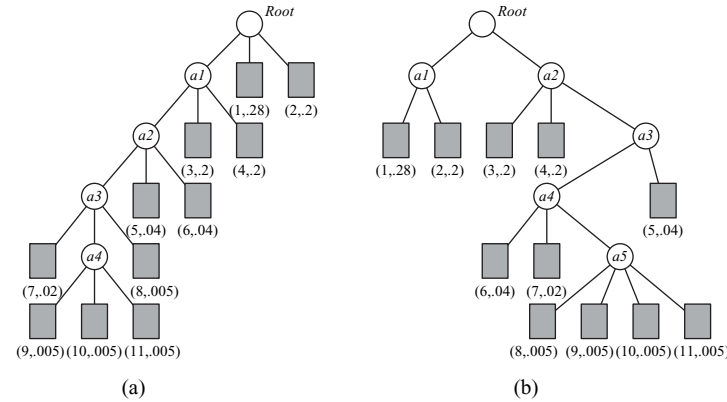


Fig. 7. Unbalanced index tree with fixed and variable fanouts; (a) index tree of fixed fanout of 3, (b) index tree of variable fanouts.

example for fixed fanouts in Figure 7(a). The index tree with variable fanout in Figure 7(b) has a better average index traversal performance than the index tree with fixed fanout in Figure 7(a).

- Multi-Attribute Air Indexing. Queries in real applications are usually specified based on multiple attributes [31]. This observation motivates the development and adoption of multi-attribute air indexing. As broadcast channels are in a linear medium, querying and broadcast scheduling for multiple attributes appear to be much more complicated. For multi-attribute indexing proposed in Reference [32], a broadcast cycle is clustered based on the most frequently accessed attribute (that is the first attribute and also called clustered 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. The second and third attributes are collectively called n*on-clustered attributes*.

  For each non-clustered attribute, a broadcast cycle can be partitioned into a number of segments called *meta-segments* [68], each of which holds a sequence of frames with non-decreasing (or non-increasing) values of that attribute. Thus, at each individual meta-segment, the data frames are clustered and the indexing techniques discussed in the previous subsection are still applicable to a meta-segment. The number of meta-segments in the broadcast cycle for an attribute is called the *scattering factor* of the attribute. The scattering factor of an attribute increases as the importance of the attribute decreases. The index tree, signature, and hybrid methods are applicable to indexing multi-attribute data frames [23]. For multi-attribute indexing, an index tree is built for each index attribute, while the signature is generated by superimposing multiple hashed attribute values.

  When two special types of queries (i.e., queries with all conjunction operators and queries with all disjunction operators) are considered, empirical comparisons show that the index tree method, although performing well for single attribute queries results in poor access time performance [23]. This is due to its large overheads for building a distributed index tree for each indexed attribute. Moreover, index tree method has an update constraint, that is, updates of a data frame are not reflected until the next broadcast cycle. The comparisons reveals that the hybrid technique is the best choice for multi-attribute queries due to its good access time and tuning time. The signature method perform close to the hybrid method for disjunction queries. The index tree method has a similar tuning time performance as the hybrid method for conjunction queries, whereas it is poor in terms of access time for any types of multi-attribute queries.

## 4. Mobile Client Caching

Caching frequently used data items in the mobile clients' local storage can improve access latency and data availability in case of disconnection. In general, cache management techniques include cache replacement, cache coherence, and cache prefetching. Cache replacement decides what data items to keep in order to maximize the space efficiency. Cache coherence considers consistency between source values and cached values of data items. Due to narrow bandwidth, data items may not be always available on air, cache prefetching loads data items in advance if a request of those items is predictable. Due to weak

connectivity, those cache management techniques adopted in wireless and mobile computing environment are different from conventional caching techniques. In the following, various cache management issues investigated in the literature are reviewed.

### 4.1. Cache Replacement

The cache replacement issue for wireless data broadcast was first studied in the Broadcast Disk project [8]. A cache replacement policy, $\mathcal{PIX}$, proposed by Acharya *et al*. selects the data item $i$ with the minimum value of $p_i/x_i$ for replacement, where $p_i$ is $i$'s access probability and $x_i$ is its broadcast frequency. Thus, an evicted item either has a low access probability or has a short retrieval delay. Since $\mathcal{PIX}$ is not an implementable policy, $\mathcal{LIX}$, a modified version of LRU is used to approximate $\mathcal{PIX}$. $\mathcal{LIX}$ maintains a number of lists: one list corresponding to each broadcast disk. A data item always enters the list corresponding to the disk in which it is broadcast. When a data item enters the cache, $\mathcal{LIX}$ evaluates $lix_i$ value only for the item $i$ at the end of each list. The $lix_i$ is determined as $p_i/x_i$. Here, the access probability $p_i$ is calculated as $p_i = \Lambda/(CurrentTime - t_i) + (1 - \Lambda)p_i$ where $t_i$ is the time of the most recent access to the item and $\Lambda$ is a control parameter ranging between 0 and 1.

Caching algorithms for Broadcast Disk systems were also studied in Reference [33]. Their work assumed that neither knowledge of future data requests nor knowledge of access probability distribution over the data items was available to the clients. The proposed *Gray* algorithm takes the factors of both access history and retrieval delay for cache replacement/prefetching into consideration. Theoretical study showed that, in terms of worst-case performance, *Gray* outperformed LRU by a factor proportional to $CacheSize/\log CacheSize$.

While most broadcast-based cache replacement schemes consider only fixed size data items, SAIU, a cache replacement scheme proposed in Reference [34] considers four factors that affects cache replacement decision of a data item $i$, including access probability ($a_i$), update frequency ($u_i$), retrieval delay ($l_i$), and data size ($s_i$). The replacement score of a data item $i$ denoted by a gain function annotated by $gain_{SAIU}(i)$ equals:

$$gain_{SAIU}(i) = \frac{l_i \cdot a_i}{s_i \cdot u_i}$$

When the space is needed to reclaim, the data item with the least gain function value is removed. Rather than access latency, stretch [12], the ratio of the access latency of a request to its service time that is defined as the ratio of the requested item's size to the bandwidth, is used to measure the performance. However, the influence of the cache consistency requirement was not considered in SAIU. Then an optimal cache replacement policy called Min-SAUD extending SAIU was investigated in Reference [35]. The revised gain function for Min-SAUD denoted by $gain_{SAUD}(i)$ is expressed as:

$$gain_{SAUD}(i) = \frac{a_i}{s_i}\left(\frac{l_i}{1 + x_i} - v\right)$$

where $x_i$ is the ratio of update rate to access rate for a data item $i$, and $v$ is the cache validation delay while the rest notations are the same as used in SAIU. In Reference [35], analytical study shows that Min-SAUD achieves optimal stretch under the standard assumptions of the independent reference model and Poisson arrivals of data access and updates. Also, the simulation result shows that Min-SAUD outperforms LRU and SAIU using on-demand broadcasts.

### 4.2. Cache Coherence

As discussed in Reference [36], a number of cache consistency models are possible for wireless and mobile computing:

- Opportunistic. A client can read any version of a data item that is available in the cache without worrying about consistency for some applications and certain defined conditions are met.
- Quasi-Caching [37]. Consistency is defined subject to application constraints that specify a tolerance of slack compared with the latest value of the source data items. For example, a cached value can be at most $x\%$ deviated from the value in the server. The cache update or invalidation occurs only when the deviation between the values exceeds the range, that is, $x\%$. Or, a query can accept old versions of data items consistent at a same snapshot [38,39].
- Latest Value. A client must always access the most recent value of a data item, so whenever the data items are updated, the corresponding cached value are needed to update or invalidate.

The former two consistency models trade consistency for performance, but it heavily depends on the application requirement. In most cases, latest value is a commonly assumed model in mobile cache management. To synchronize the client cache, a server can be either stateful or stateless. A stateful server keeps tracks of the cache content

and connection status of every client. The server directs updates to the clients as long as they are connected. However, client disconnection makes this solution hard to realize and a large client population further imposes limited scalability of this approach. Stateless server approach has greater flexibility. It does not have scalability problem since cache invalidation is performed upon broadcast and the clients are responsible to invalidate/update their cache.

The server delivers invalidation messages over a broadcast channel periodically or aperiodically. Summary of data items that are updated is called Invalidation Report (IR). From an IR, clients can decide which cached data items become invalid and if they need to refresh the invalid cached items for their queries. In Reference [40], three invalidation strategies based on the periodical broadcast are studied namely Broadcast Timestamp (BT), Amnesic Terminal (AT), and Signature (SIG).

- Broadcast Timestamp (BT). In BT algorithm, an IR is composed of the identifiers of changed data items and the timestamps of the latest change on the item happened in the last $w$ seconds. Upon receiving the IR, a client discards those cached items whose timestamps are less than the timestamps of corresponding items mentioned in the IR. In case of the age of the cache exceeds $w$ seconds, the client evacuates entire cache.
- Amnesic Terminal (AT). In AT algorithm, an IR contains the identifiers of data items that was changed since the previous IR. A client drops a cached item if it is reported in the IR. The client discards an entire cache if it missed a threshold number of consecutive IRs. AT saves more bandwidth than BT due to no inclusion of timestamps, but it requires the client constantly listens to every IR.
- Signature (SIG). In SIG algorithm, the identifiers of changed items are encoded as a *signature*. The identifiers of multiple changed items are merged as a single signature reported in an IR. A client discards a cached item when its signature matches the signature in IR. The signature is more compact in size than original item identifiers, thus saving bandwidth in IR delivery. However, it is possible to have '*false negative*' that the client discards items which are still valid.

The trade-off among the three schemes in different workload scenarios are reported in Reference [40]. Supporting those clients who may be disconnected for a long period of time causes the formation of bulky IRs. Upon an expensive wireless channel, a compact IRs is more desirable. Bit-Sequence (BS), proposed in Reference [41] is an improved version of invalidation report. BS IR consists of a sequence of bits with each bit representing a data item. The bit is set to '1' if the represented data item is updated; otherwise '0'. Each bit sequence is associated with a timestamps indicating the update time. To reduce the size of IR, updates are aggregated and a hierarchical structure of bit-sequences is formed. Given $N$ data items, an IR is then composed of $k$ ($k \leq \lfloor \log\ N \rfloor$) level BSs, that is $B_1, B_2, \ldots, B_k$. The $B_k$, the highest-level BS, has $N$ bits, representing all $N$ data items. For $B_i$, a lower level of BS, containing $2^i$ bits, each bit stands for $N/2^{k-i}$ data items. As a result, the whole IR consumes at most $2N$ bits and $k$ timestamps.

When a request is initiated by a client inside an IR interval (i.e., the duration between two consecutive IRs), the client has to wait until the IRs are received to confirm validity of the cached items. Then a long query latency is resulted. To improve the query waiting time, multiple smaller updated IRs (UIRs) proposed in Reference [42] are inserted in an IR interval. As long as the client receives UIRs and no items are invalidated, the query can be processed immediately.

### 4.3. Cache Prefetching

In the broadcast environment, data items are only available at a specific time. If a query requests an item which will only be broadcast a long time later, the satisfactory response time cannot be guaranteed. Cache prefetching is an operation to preload some data items right before any query initiation to shorten the query response time. Prefetching in a broadcast environment only requires local resources, but the cache space may be wasted if a data item is cached too early. In a Tag-team [43], a client caches data item $i$ when it is broadcast and replaces it with another data item $j$ when $j$ is broadcast. Similarly, $j$ is dropped again when $i$ is re-broadcast. Assuming that $i$ and $j$ are broadcast using a flat-disk and they are placed 180 degrees apart on a same disk, the average tag-team caching cost is one-eighth of a disk rotation, which is one-half of the cost in a demand-driven caching scheme. Thus, it doubles the performance over demand-driven caching due to the fact that the cost of a miss in Tag-team is half of the cost of a miss in demand-driven caching.

Introduced in Reference [44], $\mathcal{PT}$ algorithm is adopted on Broadcast Disk. It measures the $pt$ value of data items. The cache is used to maintain the data items with high $pt$ values. The $pt$ value is the product of the access probability ($p$) and the elapsed time ($t$) from the present time to the time when the instance of

the item to be broadcast again. Attempting to minimize average access latency, Reference [45] presented a cache update policy in which a broadcast channel is divided into time slots of equal size, that are equal to the broadcast time of a single item. Let $\lambda_i$ be the access rate for item $i$ and $\tau_i(n)$ be the amount of time from slot $n$ to the next transmission of item $i$. A time-dependent reward (latency reduction) for item $i$ at slot $n$ is given by $r(i, n) = \lambda_i \tau_i(n) + \lambda_i/2$. The proposed *W-step look-ahead* scheme made the cache update decision at slot $n$ such that the cumulative average reward from slot $n$ up to slot $n + W$ was maximized. The larger the window $W$ is, the better the access performance, the worse the complexity of the algorithm will be.

The analytical model for various prefetching policies is derived in Reference [46]. Also, in Reference [46], PP1 and PP2 policies are devised. PP1 policy achieves the minimum power consumption. PP2 policy achieve minimum access latency. Both PP1 and PP2 are static policies. For the dynamic environment where request rates are not fixed, adaptive power-aware prefetching schemes [47,48] are proposed.

## 5. Data Access Based on Application Requirements

In the previous two sections, both the discussed wireless broadcast and client caching techniques are developed for general data access. Those techniques may be quite different, subject to the nature of applications and their specific requirements. In this section, data access techniques based on the application requirements in terms of time, location, data semantics, and reliability are described. Time-critical applications demand data to be made available by a specified 'deadline,' otherwise the delayed information is of no value to the users. Location-based applications request data based on the current locations of the users. An example of such an application is to find the nearest ATM from a user's current position. Further, the semantic and reliable data access are also discussed in the following subsections.

### 5.1. Time-Critical Data Access

Timeliness of data access is of central importance in many time-critical applications. The belated information has no value to users. Take stocks as an example, belated stock price information will lead to a bad investment decision. To ensure information accessible by a specified time, the online broadcast schedules takes deadlines associated with data requests into account. Usually the schedules are applied to on-demand broadcast.

The major performance metric for time-critical data dissemination is *request drop rate*. It measures the overall performance based on the rate of number of requests which deadlines are missed to the total number of requests. A good broadcast schedule should provide the lowest request drop rate. Some developed heuristics are discussed below:

- Earliest Deadline First (EDF) [49]. Each request specifies the deadline of requested data item. The item with the earliest deadline is delivered first. This approach performs reasonably well in moderately loaded situation (i.e., the channel utilization is under 100%). In a case that multiple data items demanded by a request, $r$, the deadline of each requested items, $i$, is defined based on Equal Slack Data (EQSD) [50]:

$$deadline(i) = arrivel(r) + \frac{deadline(r) - arrival(r)}{\text{total number of items requested by } r}$$

  where $deadline(r)$ and $arrival(r)$ are the deadline and arrival time of $r$, respectively.
- Most Request First (MRF). The data items with the largest number of requests are broadcast first to satisfy most requests.
- Slack Time Inverse Number of Pending Requests (SIN-$\alpha$) [51]. The delivery priority of a data item, $i$, denoted by $sin.\alpha(i)$ is defined as

$$sin.\alpha(i) = \frac{slack(i)}{num(i)^{\alpha}}$$

  where $slack(i)$ of an item, $i$, is the remain time before the $i$'s first deadline, that is, the earliest deadline among all pending requests on $i$, $num(i)$ is the number of pending requests on $i$ and $\alpha$ is a control parameter to adjust the relative weight between $num(i)$ and $slack(i)$. In Reference [51], the theoretical bound of request drop rate is derived. In addition, a detail implementation and simulation studies are presented. Simulation result shows that SIN-$\alpha$ outperforms EDF and MRF as it can handle both deadlines and request volumes.

### 5.2. Location-Based Data Access

Location-dependent applications are unique in mobile computing environments. They access data items based on the current user locations. The requests are expressed as location-dependent queries. The most common location-dependent queries are range queries

and nearest neighbor (NN) queries. Range queries search for location-dependent data items within a distance range from a given query point or in a specified area. NN queries find data items located the closest to the query point.

To facilitate processing these location-dependent queries, the techniques for data broadcast and client caching become specialized. In particular, NN search is more complicated than range query as NN search requires backtracking when location-dependent data items are arranged in a hierarchical structure such as R-tree [52].

### 5.2.1. Location-dependent data broadcast

Although location-dependent data items can be disseminated using a multi-attribute broadcast, it is inefficient to support NN search since existing search algorithms usually require backtracking that results in scanning more than one broadcast cycle to process a single NN request. To support efficient NN searches in broadcast environments, D-tree and DSI air indexing structures were proposed.

- D-tree [53]. D-tree is a binary tree indexing a Voronoi diagram [54] in which every Voronoi cell (cell) is a region within which an NN query always finds the corresponding object as its result. An example of D-tree is shown in Figure 8. In Figure 8(a), four cells ($C_1$, $C_2$, $C_3$, and $C_4$) and six vertices ($V_1$, $V_2$, $V_3$, $V_4$, $V_5$, and $V_6$) are derived. Along a sequence of vertices, the space is recursively partitioned into two complementary subspaces. Each partitioned subspace contains about the same number of cells. The partition is along $x$- or $y$-dimension alternatively. In the figure, the whole space is first divided into $P_5$ and $P_6$. Further, $P_5$ is partitioned into $P_1$ and $P_2$ where cells are $C_1$ and $C_2$, respectively. Similarly, $P_6$ is partitioned into $P_3$ and $P_4$ representing $C_3$ and $C_4$ correspondingly. The index is then flattened and disseminated over a broadcast channel.

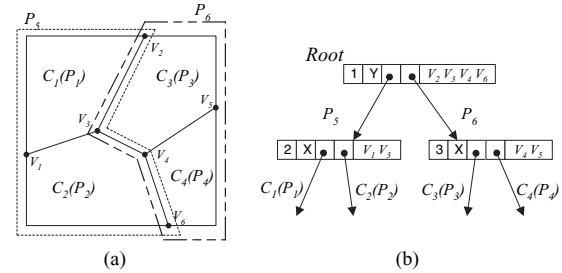


Fig. 8. D-tree; (a) division of space, (b) D-tree structure.

When a client issues an NN query (with the client position treated as a query point), it traverses the index along the path from the root to the leaf. At each level, the client follows the branch which subspace contains the query point. At the leaf, the object inside the region is the NN to the client. Other than D-tree, the subspace can be embedded as minimum bounding box and then can be indexed using R-tree. In Reference [53], D-tree is shown to outperform this R-tree based approach.

- Distributed Spatial Index (DSI) [55]. DSI distributes the index information over the whole broadcast and provides a client with sufficient information to conduct both range and NN search. The dissemination order of location-dependent data items follows the sequence of Hilbert Curve, a space filling curve. Figure 9(a) shows an Hilbert curve of order 3 covering 64 positions in which 8 data items are located and their respective HC values are 6, 11, 17, 27, 32, 40, 51, and 60. Based on the ascending HC values, data items located in the two-dimensional space are linearized and disseminated in the broadcast channel.

Figure 9(b) illustrates the broadcast structure. DSI index tables are delivered interleaving the data items.

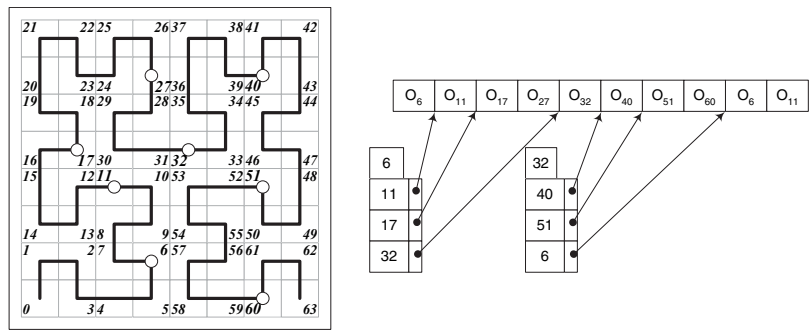


Fig. 9. Distributed spatial index; (a) Hilbert curve of order 3, (b) index table.

Each index table maintains pointers to upcoming frames (each containing one location-dependent data item). Rather than pointing all individual frames, an exponential index [56] is adopted so that the number of frames indexed is exponentially increased with the order of index table entries. Take the index table for frame $O_6$ as an example in Figure 9(b). The first pointer refers to the first upcoming frame which contains $O_{11}$. The second pointer refers to the second upcoming frame of $O_{17}$ while the third the fourth upcoming frame of $O_{32}$. Query processing on location-dependent data broadcast using HC values is discussed in Reference [57].

#### 5.2.2. Location-dependent data caching

The concept of *valid scope* is an important property of location-dependent queries. Each location-dependent query result has a valid scope that is captured as a region. Unless a client stays outside the scope, the result of the same location-dependent query remains valid. The valid scope provides client additional knowledge about the query result and saves client resources to re-execute the same query. Reference [58] roughly estimates the valid scope for a $k$-NN result by retrieving $m + k$ data items. The new $k$-NN result can be found within the previous result when the distance moved by a client from the query initiation point is not greater than $\frac{1}{2}(dist(m + k) - dist(k))$ where $dist(x)$ is the distance between the $x$th data item from the query initiation point. To exactly calculate the valid scope for a $k$-NN query as well as a window query, an online algorithm is proposed in Reference [59]. It determines the valid scope by issuing multiple TPNN [60] queries to explore influencing objects at all possible directions from existing query result.

Cache management for location-dependent queries also takes locations into consideration. In Reference [61], the distances between a current user location and cached data items are proposed as replacement decision rather than temporal-based heuristics such as Least Recently Used (LRU) and Most Recently Used (MRU). Further Away Replacement (FAR) proposed in Reference [62] is used as replacement score. The data item located farthest from the current user position is replaced first if space is needed to accept new data items. In Reference [63], replacement score is based on both the distance and valid scope area of a cached data item. Valid scope area is considered due to an intuition that the larger the valid scope area of the data item is, the higher the access probability of it will be. Then two policies are proposed namely probability area (PA) and and probability area inverse distance (PAID). By PA, the cost function of an item, $i$, $c^{\mathrm{PA}}(i)$ is expressed as $P_i \cdot A(v_i)$ and by PAID, the cost function of an item $i$, $c^{\mathrm{PAID}}(i)$ is $P_i \cdot A(v_i)/dist(i)$ where $P_i$ is the access probability of item $i$, $A(v_i)$ is the area of valid scope of $i$, and $dist(i)$ is the distance of $i$ from a current client position. In Reference [63], through simulation, both PA and PAID policies provide substantial performance improvement over LRU and FAR policies. In Reference [64], combined temporal and distance factors in determining item relevance are discussed. In addition, in Reference [65], the part of underlying index structures are cached in addition to cached data items to support various kind of location-dependent queries.

### 5.3. Semantic-Based Data Access

Due to weak connectivity, clients should be made more aware of the cache content. If a client can determine that all data items needed to answer its queries are available in its cache, many requests can be served locally without contacting the server. To provide clients such ability, a semantic caching is proposed in References [61,66,67]. The cached items are grouped and collectively described as semantic regions using query expressions. When a new query is determined with query containment [68] if it is contained by a semantic region, a client can assure that no additional data item(s) is required from the server. In case a new query is partially covered, a remainder query is formulated to fetch missing data from the server.

With the same rationale, a semantic broadcast scheme is derived. Due to limited bandwidth, not all data items are broadcast on air. With semantic information present in the index, the clients can query required data items and determine if the missing parts for their queries are needed. In Reference [69], that data items are clustered based on semantics in different granularity are discussed.

### 5.4. Reliable Data Access

Wireless communication is error-prone. Data might be corrupted or lost due to factors such as signal interference. When an error occurs, mobile clients have to wait for the next copy of the data if no special precaution is taken. This will increase the access latency. To deal with unreliable wireless communication, an idea is to introduce controlled redundancy in the broadcast schedule. Such redundancy allows mobile clients to obtain their data items from the broadcast cycle even in

the presence of errors. This eliminates the need to wait for the next broadcast of the data item whenever any error occurs. In References [70,71], additional index information is distributed in the broadcast. Clients can resume searching right after recovery without waiting for next broadcast cycle. To have a reliable data delivery, a set of $k$ data blocks are encoded using Tornado codes, proposed in Reference [72], into a set of $n$ encoded blocks (where $n$ is a small multiple of $k$). A client can collect any $k$ out of $n$ encoded blocks to decode and determine the original data blocks. Similar idea can be found in Reference [73] where Adaptive Information Dispersal Algorithm (AIDA) is adopted to distribute the content of $m$ data items in $n$ ($n > m$) frames. By collecting a certain amount of frames, the original data items can be reconstructed by using a reconstruction transmission matrix. By the same algorithm, security is also supported that only legal clients know the reconstruction transformation matrix to recover the original data items.

## 6. Summary

This survey paper reviewed wireless data broadcast and mobile caching techniques in support of efficient pervasive data access in wireless and mobile computing environments. A number of broadcast scheduling and air-index schemes appeared in the literature to address performance requirements in access and energy efficiency. Push-based, on-demand, and hybrid broadcast scheduling techniques were discussed. Push-based broadcast is attractive when access patterns are known in advance while on-demand broadcast is preferable for dynamic access pattern. Hybrid data broadcast offers more flexibility by combining both strength of push-based and on-demand broadcasts. Several air indexing techniques, based on hashing, index tree, signature and hybrid of the above, were described. Air index in support of multi-attribute queries were also discussed.

To improve access latency and provide higher data availability, client data caching plays an important role. To optimize the cache performance, cache management issues such as cache prefetching, cache invalidation, and cache replacement were discussed. For cache invalidation, different kinds of broadcast-based invalidation reports are described. In addition, cache replacement policies are also reviewed. Finally, to address specific application requirements in time, location, data semantics and reliability, data broadcast and mobile caching schemes have been developed. This survey paper also provides a good introduction of those studies.

## References


1. *General Packet Radio Services (GPRS)*. http://www.topology.org/comms/gprs.html.
2. *3rd Generation Partnership Project 2*. http://www.3gpp2.org.
3. Kotz D, Essien K. Analysis of a campus-wide wireless network. In *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking (Mobicom), Atlanta, GA, USA, September 23–28* 2002, pp. 107–118.
4. *Wi-Fi Alliance*. http://www.weca.net/OpenSection/pdf/Wi-Fi_Protected_Access_Overview. pdf.
5. *IEEE 802.11, The Working Group Setting the Standards for Wireless LANs*. http://grouper.ieee.org/groups/802/11/index.html.
6. Su M, Chi C-H. Architecture and performance of application networking in pervasive content delivery. In *Proceedings of the 21st International Conference on Data Engineering (ICDE), Tokyo, Japan, April 5–8*, 2005, pp. 656–667.
7. Wong JW. Broadcast Delivery. *Proceedings of the IEEE* 1988; **76**(12): 1566–1577.
8. Acharya S, Alonso R, Franklin MJ, Zdonik SB. Broadcast disks: Data management for asymmetric communications environments. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, CA, USA, May 22–25*, 1995, pp. 199–210.
9. Hameed S, Vaidya NH. Efficient Algorithms for Scheduling Data Broadcast. *ACM/Baltzer Journal of Wireless Networks (WINET)* 1999; **5**(3): 183–193.
10. Su C-J, Tassiulas L, Tsotras VJ. Broadcast Scheduling for Information Distribution. *ACM/Baltzer Journal of Wireless Networks (WINET)* 1999; **5**(2): 137–147.
11. Vaidya NH, Hameed S. Scheduling Data Broadcast in Asymmetric Communication Environment. *ACM/Baltzer Journal of Wireless Networks (WINET)* 1999; **5**(3): 171–182.
12. Acharya S, Muthukrishnan S. Scheduling on-demand broadcasts: New metrics and algorithms. In *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom), Dallas, Texas, USA, October 25–30*, 1998, pp. 43–54.
13. Aksoy D, Franklin MJ. R × W: A Scheduling Approach for Large-Scale On-Demand Data Broadcast. *IEEE/ACM Transaction on Networking* 1999; **7**(6): 846–860.
14. Hu Q, Lee DL, Lee W-C. Performance evaluation of a wireless hierarchical data dissemination system. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom), Seattle, Washington, USA, August 15–19*, 1999, pp. 163–1739.
15. Dykeman HD, Ammar MH, Wong JW. Scheduling algorithms for videotex systems under broadcast delivery. In *Proceeding of International Conference on Communications, Toronto, Canada*, 1988.
16. Zipf GK. *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*. Addison-Wesley: Reading, MA, 1949.
17. Datta A, VanderMeer DE, Celik A, Kumar V. Broadcast Protocols to Support Efficient Retrieval from Databases by Mobile Users. *ACM Transactions on Database Systems (TODS)* 1999; **24**(1): 1–79.
18. Imielinski T, Viswanathan S, Badrinath BR. Data on Air: Organization and Access. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 1997; **9**(3): 353–372.
19. Wong JW, Dykeman HD. Architecture and performance of large scale information delivery networks. In *Proceedings of International Teletraffic Congress, Torino, Italy*, 1988.

20. Acharya S, Franklin MJ, Zdonik SB. Balancing push and pull for data broadcast. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data, Tucson, AZ, USA, USA, May 13-15*, 1997, pp. 183–194.
21. Hu C-L, Chen M-S. Dynamic data broadcasting with traffic awareness. In *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS), Vienna, Austria, July 2–5*, 2002, pp. 112–119.
22. Jiang S, Vaidya NH. Scheduling data broadcast to 'Impatient' users. In *Proceedings of the ACM International Workshop on Data Engineering for Wireless and Mobile Access, Seattle, WA, USA, August 20*, 1999, pp. 52–59.
23. Lin C-W, Lee DL. Adaptive data delivery in wireless communication environments. In *Proceedings of the 20th International Conference on Distributed Computing Systems (ICDCS), Taipei, Taiwan, April 10–13*, 2000, pp. 444–452.
24. Stathatos K, Roussopoulos N, Baras JS. Adaptive data broadcast in hybrid networks. In *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB), Athens, Greece, August 25–29*, 1997, pp. 326–335.
25. Imielinski T, Viswanathan S, Badrinath BR. Power efficient filtering of data an air. In *Proceedings of the 4th International Conference on Extending Database Technology (EDBT), Cambridge, UK, March 28–31*, 1994, pp. 245–258.
26. Lee W-C, Lee DL. Using Signature Techniques for Information Filtering in Wireless and Mobile Environments. *Distributed and Parallel Databases* 1996; **4**(3): 205–227.
27. Yang X, Bouguettaya A. Broadcast-Based Data Access in Wireless Environments. In *Proceedings of the 8th International Conference on Extending Database Technology (EDBT), Prague, Czech Republic, March 25–27*, 2002, pp. 553–571.
28. Hu Q, Lee W-C, Lee DL. A Hybrid Index Technique for Power Efficient Data Broadcast. *Distributed and Parallel Databases* 2001; **9**(2): 151–177.
29. Chen M-S, Yu PS, Wu K-L. Indexed sequential data broadcasting in wireless mobile computing. In *Proceedings of the 17th International Conference on Distributed Computer Systems (ICDCS), Baltimore, Maryland, USA, May 28–30*, 1997, pp. 124–131.
30. Shivakumar N, Venkatasubramanian S. Energy-Efficient Indexing for Information Dissemination in Wireless Systems. *ACM/Baltzer Journal of Mobile Networks and Nomadic Applications (MONET)* 1996; **1**(4): 433–446.
31. Zhang J, Gruenwald L. Optimizing data placement over wireless broadcast channel for multi-dimensional Range Query Processing. In *Proceedings of 5th IEEE International Conference on Mobile Data Management (MDM 2004), Berkeley, CA, USA, January 19–22*, 2004, pp. 256–265.
32. Hu Q, Lee W-C, Lee DL. Power conservative multi-attribute queries on data broadcast. In *Proceedings of the 16th International Conference on Data Engineering (ICDE), San Diego, CA, USA, February 28–March 3*, 2000, pp. 157–166.
33. Khanna S, Liberatore V. On Broadcast Disk Paging. *SIAM Journal of Computing* 2000; **29**(5): 1683–1702.
34. Xu J, Hu Q, Lee DL, Lee W-C. SAIU: An efficient cache replacement policy for wireless on-demand broadcasts. In *Proceedings of the 2000 ACM International Conference on Information and Knowledge Management (CIKM), McLean, VA, USA, November 6–11*, 2000, pp. 46–53.
35. Xu J, Hu Q, Lee W-C, Lee DL. Performance Evaluation of an Optimal Cache Replacement Policy for Wireless Data Dissemination. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 2004; **16**(1): 125–139.
36. Acharya S, Franklin MJ, Zdonik SB. Disseminating Updates on Broadcast Disks. In *Proceedings of the 22th International Conference on Very Large Data Bases (VLDB), Mumbai (Bombay), India, September 3–6*, 1996, pp. 354–365.
37. Alonso R, Barbará D, Garcia-Molina H. Data Caching Issues in an Information Retrieval System. *ACM Transactions Database Systems (TODS)* 1990; **15**(3): 359–384.
38. Pitoura E, Chrysanthis PK. Exploiting versions for handling updates in broadcast disks. In *Proceedings of the 25th International Conference on Very Large Data Bases (VLDB), Edinburgh, Scotland, UK, September 7–10*, 1999, pp. 114–125.
39. Pitoura E, Chrysanthis PK. Multiversion Data Broadcast. *IEEE Transactions on Computers* 2002; **51**(10): 1224–1230.
40. Barbará D, Imielinski T. Sleepers and workaholics: Caching strategies in mobile environments. In *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, Minneapolis, MN, USA, May 24–27*, 1994, pp 1–12.
41. Jing J, Elmagarmid AK, Helal A, Alonso R. Bit-Sequences: An Adaptive Cache Invalidation Method in Mobile Client/Server Environments. *ACM/Baltzer Journal of Mobile Networks and Nomadic Applications (MONET)* 1997; **2**(2): 115–127.
42. Cao G. A Scalable Low-Latency Cache Invalidation Strategy for Mobile. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 2003; **15**(5): 1251–1265.
43. Acharya S. *Broadcast Disks: Dissemination-based Management for Assymmetic Communication Environments.* PhD thesis, Brown University, 1998.
44. Acharya S, Franklin MJ, Zdonik SB. Prefetching from Broadcast Disks. In *Proceedings of the 12th International Conference on Data Engineering (ICDE), New Orleans, Louisiana, USA, February 26–March 1*, 1996, pp. 276–285.
45. Tassiulas L, Su C-J. Optimal Memory Management Strategies for a Mobile User in a Broadcast Data Delivery System. *IEEE Journal on Selected Areas in Communication* 1997; **15**(7): 1226–1238.
46. Grassi V. Prefetching policies for energy saving and latency reduction in a wireless broadcast data delivery system. In *Proceedings of the 3rd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM), Boston, MA, USA, August 20*, 2000, pp. 77–84.
47. Hu H, Xu J, Lee DL. Adaptive power-aware prefetching schemes for mobile broadcast environments. In *Proceedings of International Conference on Mobile Data Management (MDM), Melbourne, Australia, January 21–24*, 2003, pp. 374–380.
48. Yin L, Gao G. Adaptive Power-Aware Prefetch in Wireless Networks. *IEEE Transactions on Wireless Communications* 2004; **3**(5): 1648–1658.
49. Xuan P, Sen S, González O, Fernandez J, Ramamritham K. Broadcast on demand: Efficient and timely dissemination of data in mobile environments. In *Proceedings of IEEE Real Time Technology and Applications Symposium (RTAS), Montreal, Canada, June 9–11*, 1997, pp. 38–48.
50. Lam K-Y, Chan E, Yuen JC-H. Data broadcast for time-constrained read-only transactions in mobile computing systems. In *Proceedings of International Workshop on Advance Issues of E-Commerce and Web-based Information Systems, Santa Clara, CA, April 8–9*, 1999.
51. Xu J, Tang X, Lee W-C. Time-Critical On-Demand Data Broadcast: Algorithms, Analysis, and Performance Evaluation. *IEEE Transactions on Parallel and Distributed Systems* 2006; **17**(1): 3–14.
52. Roussopoulos N, Kelley S, Vincent F. Nearest neighbor queries. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, CA, USA, May 22–25*, 1995, pp. 71–79.
53. Xu J, Zheng B, Lee W-C, Lee DL. The D-Tree: An Index Structure for Planar Point Queries in Location-Based Wireless Services. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 2004; **16**(12): 1526–1542.
54. Aurenhammer F. Voronoi Diagrams—A Survey of a Fundamental Geometric Data Structure. *ACM Computing Survey* 1991; **23**(3): 345–405.
55. Lee W-C, Zheng B. DSI: A fully distributed spatial index for location-based wireless Broadcast. In *Proceedings of the 25th*


 

*International Conference on Distributed Computing Systems (ICDCS), Columbus, OH, USA, June 6–10*, 2005.
56. Xu J, Lee W-C, Tang X. Exponential Index: A parameterized distributed index scheme for data on air. In *Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services (Mobisys), Boston, MA, USA, June 6–9*, 2004, pp. 153–164.
57. Zheng B, Xu J, Lee W-C, Lee DL. Energy-Conserving air indexes for nearest neighbor search. In *Proceedings of the 9th International Conference on Extending Database Technology (EDBT), Heraklion, Crete, Greece, March 14–18*, 2004, pp. 48–66.
58. Song Z, Roussopoulos N. K-Nearest neighbor search for moving query point. In *Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases (SSTD), Redondo Beach, CA, USA, July 12–15*, 2001, pp. 79–96.
59. Zhang J, Zhu M, Papadias D, Tao Y, Lee DL. Location-based spatial queries. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, San Diego, CA, USA, Junuary 9–12*, 2003, pp. 443–454.
60. Tao Y, Papadias D. Time-parameterized queries in spatio-temporal databases. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, USA, June 3–6*, 2002, pp. 334–345.
61. Dar S, Franklin MJ, Jónsson BT, Srivastava D, Tan M. Semantic data caching and replacement. In *Proceedings of the 22th International Conference on Very Large Data Bases (VLDB), Mumbai (Bombay), India, September 3–6*, 1996, pp. 330–341.
62. Ren Q, Dunham MH. Using semantic caching to manage location dependent data in mobile computing. In *Proceedings of the 6th ACM Annual International Conference on Mobile Computing and Networking (Mobicom), Boston, MA, USA, August 6–11*, 2000, pp. 210–221.
63. Zheng B, Xu J, Lee DL. Cache Invalidation and Replacement Strategies for Location-Dependent Data in Mobile Environments. *IEEE Transactions on Computers* 2002; **51**(10): 1141–1153.
64. Xu B, Ouksel AM, Wolfson O. Opportunistic resource exchange in inter-vehicle ad-hoc networks. In *Proceedings of 5th IEEE International Conference on Mobile Data Management (MDM), Berkeley, CA, USA, January 19–22*, 2004, pp. 4–12.
65. Hu H, Xu J, Wong WS, Zheng B, Lee DL, Lee W-C. Proactive caching for spatial queries in mobile environments. In *Proceedings of the 21st International Conference on Data Engineering (ICDE), Tokyo, Japan, April 5–8*, 2005, pp. 403–414.
66. Lee KCK, Leong HV, Si A. Semantic Query Caching in a Mobile Environment. *Mobile Computing and Communication Review ($MC^2R$)* 1999; **3**(2): 28–36.
67. Ren Q, Dunham MH, Kumar V. Semantic Caching and Query Processing. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 2003; **15**(1): 192–210.
68. Chekuri C, Rajaraman A. Conjunctive query containment revisited. In *Proceedings of the 6th International Conference on Database Theory (ICDT), Delphi, Greece, January 8–10*, 1997, pp. 56–70.
69. Lee KCK, Leong HV, Si A. Semantic Data Broadcast for a Mobile Environment Based on Dynamic and Adaptive Chunking. *IEEE Transactions on Computers* 2002; **51**(10): 1253–1268.
70. Lo S-C, Chen ALP. An Adaptive Access Method for Broadcast Data under an Error-Prone Mobile Environment. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 2000; **12**(4): 609–620.
71. Tan K-L, Ooi BC. On Selective Tuning in Unreliable Wireless Channels. *Data and Knowledge Engineering* 1998; **28**(2): 209–231.
72. Byers JW, Luby M, Mitzenmacher M, Rege A. A digital fountain approach to reliable distribution of bulk data. In *Proceeding of the ACM SIGCOMM'98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, Vancouver, B.C., Canada, August 31–Sep 4*, 1998, pp. 56–67.
73. Baruah SK, Bestavros A. Pinwheel scheduling for fault-tolerant broadcast disks in real-time database systems. In *Proceedings of the 13th International Conference on Data Engineering (ICDE), Birmingham, U.K. April 7–11*, 1997, pp 543–551.

## Authors' Biographies

**Ken C. K. Lee** is currently a Ph.D. candidate at the Department of Computer Science and Engineering, Pennsylvania State University. He received his Bachelor and MPhil degrees in Computing from the Hong Kong Polytechnic University in 1996 and 1999, respectively. Before joining Pennsylvania State University, he was a research assistant at the Department of Computing, the Hong Kong Polytechnic University during 2002–2004 and he was a system engineer in the Center of Innovation and Technology, the Chinese University of Hong Kong during 1999–2002. His research interest include data management in wireless and mobile computing environment, spatio-temporal databases, and location-based services. He is now a member of IEEE Computer Society.

**Wang-Chien Lee** is an associate professor of Computer Science and Engineering at Pennsylvania State University. He received his B.S. from the Information Science Department, National Chiao Tung University, Taiwan, his M.S. from the Computer Science Department, Indiana University, and his Ph.D. from the Computer and Information Science Department, the Ohio State University. Prior to joining Penn State, he was a principal member of the technical staff at Verizon/GTE Laboratories, Inc. Dr Lee performs cross-area research in database systems, pervasive/mobile computing, and networking. He is particularly interested in developing data management techniques (including accessing, indexing, caching, aggregation, dissemination, and query processing) for supporting complex queries in a wide spectrum of networking and mobile environments such as peer-to-peer networks, mobile ad-hoc networks, wireless sensor networks, and wireless broadcast systems. He has served as a guest editor for several journal special issues on mobile database-related topics, including *IEEE Transaction on Computer*, *IEEE Personal Communications Magazine*, *ACM MONET*, and *ACM WINET*. He was the founding program committee co-chair for the International Conference on Mobile Data Management. He is a member of the IEEE and the Association for Computer Machinery.

**Sanjay Kumar Madria** received his Ph.D. in Computer Science from Indian Institute of Technology, Delhi, India in 1995. He is an assistant professor, Department of Computer Science, at University of Missouri-Rolla, USA. Earlier he was a visiting assistant professor in the Department of Computer Science, Purdue University, West Lafayette, USA. He has also held appointments at Nanyang Technological University in Singapore and University Sains Malaysia, Malaysia. He has published more than 100 journal and conference papers in the areas of web data warehousing, mobile databases, nested transaction management, and performance issues. He guest edited WWW Journal and Data and Knowledge Engineering Sp. Issues on Web data management and Data warehousing. He was Program Chair for EC&WEB 00&01 conferences held, UK and Germany and workshop chair for 'Internet Data Management' workshop at Florence, Italy held in September 1999. He was PC Chair of Secure and Reliable Mobile workshop held in New Orleans, October 2001. He is serving as PC member of various database conferences and workshops and reviewer for many reputed database journals such as IEEE TKDE, IEEE Computer. Dr Madria has given tutorials on mobile databases and web warehousing in many international conferences and has received excellent response. He is regular invited panelist in NSF and Sweden Council of Research. He was invited keynote speaker in Annual Computing Congress in October 99 in Canada and invited speaker in Conference on Information Technology, 2001. He received DEXA award for his contribution to E-commerce and Web Data Management research. His research is supported by grants from NSF, DOE, UM research board and a grant from industry. He is a IEEE Senior Member.