Journal of Computing and Information Technology - CIT 16, 2008, 3, 197–207
doi:10.2498/cit.1001112

A Hashing Scheme for Multi-channel Wireless Broadcast

Muthuswamy Vijayalakshmi and Arputharaj Kannan

Department of CSE, College of Engineering, Anna University, India

The rapid development of wireless communication technology and battery-powered portable devices is making mobile information services increasingly popular. Since the bandwidth resource of wireless networks is scarce and the mobile devices have a limited battery capacity, any solution for information access must be devised in such a way that time and power consumption for the devices are minimized. Data broadcast is a promising technique to improve the bandwidth utilization and conserve the power consumption in a mobile computing environment. This paper proposes a hashing scheme for information access via wireless broadcast through multiple channels in which hash functions are used to index broadcast information across multiple channels. In this scheme, two different hash functions called Primary Hash Function (PHF) and Secondary Hash Function (SHF) are used, where PHF is used to determine the channel in which the desired data item is to be broadcasted and SHF is used to locate the data item within that channel. The proposed hashing scheme reduces both the access latency and tuning time and shortens the broadcast length. Moreover, Access Probabilities of data items and User Profiles that indicate the client behavior in the environment at any given time are considered in this system to construct an efficient broadcast schedule. This broadcast schedule is a non-flat data broadcast that further reduces the average access latency. Finally, Caching techniques are also implemented to further improve the access latency and tuning time.

Keywords: mobile computing, wireless data broadcast, data dissemination, hashing

1. Introduction

Mobile computing is the result of the convergence of high-speed wireless networks and personal mobile devices. In the near future, it is anticipated that a massive number of mobile users, carrying portable devices will be able to access a variety of information from anywhere and at any time. The ability of mobile users to move and access information ubiquitously has opened new classes of data applications, which promise to make our society more efficient and our lives more sophisticated. For example, alert systems can notify a mobile client when a registered alert fires (e.g., the stock price grows higher than certain pre-set threshold); and mobile users can query location-dependent information (e.g., nearest restaurant) based on their current locations. Without doubt, mobile computing is becoming an important part of our daily life. However, the major limitations in a mobile environment such as limited bandwidth for communication, limited power supply, frequent disconnections, unrestricted mobility and limited client capacities become serious concerns that are prevalent in this environment. These limitations affect the methods used to disseminate information to mobile clients.

Data broadcast is an attractive data dissemination method in a mobile computing environment. With the ***broadcast scheme***, the server repeatedly broadcasts all the information at regular intervals. This scheme is very attractive since a single broadcast of a data item can satisfy the entire outstanding request for that data item simultaneously. As such, broadcast can scale up to an arbitrary number of users. Also, data broadcast can take advantage of the large downlink capacity available to the server when delivering information to mobile clients. There are two different approaches to broadcasting data. The normal method is the one where the server does not provide any auxiliary information about the broadcast. Hence, in this method a client may have to access all the data items in a broadcast cycle to retrieve the desired data item. This requires the mobile client to listen to

the broadcast all the time, which is inefficient since clients have a scarce power supply.

On the other hand, a more efficient method, called ***indexed broadcast*** has been used in [11] in which the server transmits index information about the broadcast before the actual broadcast. In this method, the client first reads the index and finds out when the required data would arrive in the broadcast, dozes until then before waking up to read the data. Hence in this broadcast scheme, tuning time is obviously less than the normal scheme. But the drawback of this solution is that broadcast cycles are lengthened due to the transmission of additional index information leading to an increase in access latency. One of the solutions provided to this problem uses a tree structure to represent the index information in which data is kept in the leaves and the tree is traversed from the root following the index information kept in the nodes [5]. Moreover, the structure is distributed in the form of tables as discussed in [6], which suits the sequential-access broadcast environment. In few other systems, the index has been kept as a simple hash function [4], [1] which takes up constant space and thus reduces the length of the broadcast cycle. These Hash functions allow a simplified design at the server side.

In a mobile computing environment, two parameters are typically used to measure the effectiveness of any scheme for data dissemination. One, ***access latency*** that refers to the period of time elapsed from the moment a mobile client issues a query to the moment when the requested data is received by the client and two, ***tuning time*** that refers to the period of time spent by the client staying active in order to obtain the requested data. In this work, we focus on designing and implementing new techniques to reduce both access latency and tuning time by broadcasting data over multiple channels. Broadcast is a well known way of scalably transmitting data to multiple clients. But this broadcast suffers high response time due to the sequential nature of data access. One solution provided in the literature for this problem is to increase the available bandwidth by increasing the number of broadcast channels [12]. Hence, in this paper we propose a new framework that allows fast access to data that are broadcast over multiple channels by increasing the number of broadcast channels.

The scheme proposed in this paper aims at solving the problem of broadcasting a set of data items across a set of broadcast channels in such a way that it reduces both the access latency and tuning time in order to save time and power for mobile clients. We use Hash Functions to index the broadcast information in order to reduce the *tuning time* when data are broadcasted across multiple channels. Access Probabilities of data items are used as an indicator of the popularity of data items and User Profiles [8] are used to indicate the interests of mobile clients. We use these two factors to analyze clients' behavior in the mobile environment, which is considered to construct an efficient non-flat data broadcast schedule. This non-flat data broadcast reduces the *access latency*. In this system, caching techniques are also implemented in mobile clients to reduce the time for answering a query. Moreover, the availability of cache in the mobile unit further improves the access latency and tuning time of the system. Finally, mobility of the clients and handoff are also handled appropriately in this system.

The rest of this paper is organized as follows. Section 2 provides a survey of related works and its salient features. It also highlights the work presented in this paper. Section 3 describes the system used in designing efficient access broadcast programs. In Section 4, we discuss the proposed system architecture with the add-ons in the normal mobile computing architecture. Section 5 provides the simulation of the work and the results obtained. Section 6 gives a conclusion on this work and suggests some possible enhancements.

2. Related Works

Several related research issues have attracted a number of researchers to pay a considerable amount of attention to on-demand broadcast, data indexing, access frequencies estimation and client cache management.

Imielinski et al. [11] aimed to solve the problem of organizing data on wireless networks in order to provide fast and low power access to users equipped with mobile devices by proposing a tree-based indexing scheme for data broadcast. They also proposed an indexing scheme called

Distributed indexing which is an improved version of the previous method. With distributed indexing the index replications are cut down since it is sufficient to have only the index *relevant* to the data immediately following it in the broadcast. But this index information consumes a lot of space that can be efficiently used for transmission of data items. Access latency and tuning time are two conflicting performance measures that cannot be minimized at the same time with these methods. Hence it is necessary to provide a better scheme for effective dissemination of data in wireless broadcasts, that minimizes both access latency and tuning time.

Jianliang Xu et al. [3] proposed a method to make the indexing scheme tunable, in which any one of the parameter(s) is limited to a certain bound and the other is minimized. They used a linear, but distributed index instead of a tree-based one. The structure of the exponential index proposed by them allows searching to start anywhere in the index and also to recover from any error quickly, thus providing error resilience. This scheme is tunable, that is, the index can be tuned in such a way to optimize either access latency or tuning time. Error resilience is also achieved with their method by quickly recovering from link errors due to the distributed structure of the indexing scheme. But the partitioning of the indexing space needs further optimization. Moreover, the issues of balancing both access latency and tuning time are not considered in this paper.

Imielinski et al. [10] also proposed two hashing protocols for indexing the broadcast. These techniques are improved by Yuxia Yao et al. [13] and used for non-flat broadcast in a more efficient manner. However, the hash scheme used by them is not fully hole free and they provide a minimal collision resolving mechanism. The proposed scheme called MHash in [13] simultaneously reduces both access latency and tuning time in wireless broadcast. Besides, using hash function eliminates the need to broadcast index structures and thus shortens the broadcast cycles. Popular items are broadcasted frequently, thereby reducing access latency. This naturally leads to less tuning time for these items, but their scheme works only for a single-channel environment.

In the literature that deals with broadcasting, indexing schemes are discussed only for a single channel environment. Navathe et al. [12] proposed using multiple channels effectively in order to improve the performance. Multiple channels mean that many data items are available in the broadcast at any point of time, possibly serving multiple clients. But the authors assume that the client is capable of listening to only one channel at a time. Hopping is required to read data from another channel and this is assumed to take constant time. Even though there are multiple channels, each data item is still unreplicated. In order to overcome these problems, we take complement of the hashing scheme in which we apply it for multiple channels so that tuning time, access latency and broadcast length are optimized in our system.

Caching is an important issue in mobile environment since it leads to less power consumption by mobile devices. Cache Invalidation is of importance and different in this environment due to the unrestricted mobility of mobile clients and dynamic nature of the information that the mobile devices try to access. Existing schemes such as Timestamps, Bit Sequences [2] are all too complex and the size of the invalidation report is too large. Joe Chun-Hung Yuen et al. [4] proposed to use the real time characteristic of data items and the average life span of data items to invalidate the cache. The scheme is referred to as Invalidation by Absolute Validity Interval (IAVI) in which each item has an associated Absolute Validity Interval (AVI) that indicates its average lifetime. Two time intervals are used in this method. One, False Valid Period (FVP) is the period where AVI overestimates the validity period and causes the mobile unit to read the data that might be outdated. Two, False Invalid Period (FIP) is when AVI underestimates the validity period. This makes the mobile client discard his cache entry sooner than needed. By suitably adjusting the AVI based on update intervals, the values of FVP and FIP are kept low. The advantage of this scheme is that Implicit Invalidation is performed by checking the cached items when they are referenced. Explicit Invalidation is done by reading the Invalidation Report that is broadcasted by the server when a database update causes the Absolute Validity Interval (AVI) of a data item to decrease. Thus, in their method, the size of the Invalidation Report is much smaller than the earlier schemes. As our work also aims at the reduction of power consumption, we have also implemented this

caching scheme in our system that leads to low power consumption by mobile clients. In addition, we perform the temporal reasoning tasks explanation and prediction on these intervals using Allen's Interval Algebra whenever it is necessary [9].

In broadcast systems, the server is unable to learn about how the information is used, which data items are popular among clients etc. Such knowledge would however help in an efficient construction of broadcast schedules. Petros Nicopolitidis et al. [7] proposed an adaptive push system, which uses a learning automaton at the broadcast server. After an item is broadcasted, each client waiting for this item acknowledges the receipt via transmission of a short, power-controlled feedback pulse. These acknowledging pulses add up at the server, and the automaton uses the strength of the received pulse to update the probability estimated. These converge to the actual demand probability. The advantage of this simple scheme is that it judges the behavior of the clients in the environment. On the other hand, this scheme assumes that items are broadcasted one at a time and the server has to wait until it gets the response pulses before it broadcasts the next item and all response pulses are assumed to have the same signal strength. But, in this paper we introduce a new learning mechanism in which a separate channel is allocated for pull requests and the user responds with response pulses that are used to calculate demand probability apart from users' interests and profiles.

Compared with all the works found in the literature, our work is different and effective in many ways. First, we propose a new hashing scheme by extending the Hash Functions explained by Yuxia Yao et al. [13] to index broadcast information across multiple channels. Second, we consider access probabilities of data items to determine the popularity of data items and user profiles to extract and analyze history and thus track users' preferences and activity patterns in specific contexts. These two factors are introduced to predict the behavior of clients present in the environment at any given time and are used for constructing efficient broadcast schedule. Third, caching techniques proposed by Joe Chun-Hung Yuen et al. [4] are implemented to improve access latency and tuning time and, in addition, we provide explanation and prediction features. Finally, the unrestricted mobility enjoyed by mobile clients is handled by transferring their profiles automatically between base stations and by invalidating the cached data items that vary with location.

3. Designing Efficient Access Broadcast Programs

In this system, efficient access broadcast programs are designed that index the information using hash functions over multiple channels. Two hash functions, namely Primary Hash Function (PHF) and Secondary Hash Function (SHF) are designed for this purpose. Primary Hash Function (PHF) determines the channel in which the interested item will be broadcasted and Secondary Hash Function (SHF) determines the position of the item in that channel. Items that are popular in a cell are more readily accessible than the ones that the clients are not much interested in, because they are replicated to produce a non-flat broadcast. Mobile clients can tune into the broadcast schedule to obtain the results of queries from the application. If the data item gets corrupted in transmission, the clients can use the replicas. If broadcast does not contain the requested data, they can send explicit pull requests to the server and obtain the results. Data items are cached in the mobile unit in order to reduce the time and power spent in getting the results of queries. Database updates cause information that is broadcasted earlier and stored in the cache to become stale. This is handled by the server using Invalidation Reports broadcast.

3.1. Broadcast Cycle Construction

Broadcast Cycle Constructor selects items to be broadcasted not only based on their profile and access probabilities, but also with current popularity. At the end of every broadcast cycle, this component selects a predetermined number of items based on their access probabilities, which is computed using the algorithm discussed in section 3.2. Items with the largest number of pull requests select themselves for the next broadcast, even if they have low probability. For instance, cricket scores may have a low probability, but many users can start querying it suddenly. To avoid explicit pull requests

that will overload the server, the Broadcast Cycle Constructor selects items with the largest number of pull requests for the next broadcast.

A single broadcast cycle contains a number of slots each of which has the following fields,

- Slot ID – a unique identifier for the slot.
- Key – 32 bit key of the data item.
- Hash Functions – the PHF and SHF that are to be used by mobile clients.
- Data – the real information.
- Last Update Time – the time of last update of the data item.
- AVI – Absolute Validity Interval, the expected validity period of the item.
- Mobility Effect – a flag that tells whether the data is mobility dependent.
- Distance Pointer – a pointer that is used to follow chained items.
- Replication Bound – the maximum number of replications allowed for any item in the broadcast.
- Broadcast Cycle Length – length of the current broadcast cycle, in number of slots.
- Checksum – CRC32 checksum of all the above fields.

The steps followed in the construction of a single broadcast are:

Step (i). For each selected item, the Primary Hash Function (PHF) is applied to compute the channel in which the item is broadcasted.

$$H(k) = k \mod Nc. \tag{1}$$

— k – Key of the Data item.
— Nc – Number of broadcast channels.

Step (ii). There are n channels namely $C_1 \ldots C_n$, where C_i, is the i^{th} channel that contains the replication of the same data item found in channel C_1. The replication frequency of each item is computed using the formulae,

$$C_1 = 1, \tag{2}$$

$$C_i = \left\lceil \frac{r_i}{r_1} - \frac{1}{2} \right\rceil, \; 2 \leq i \leq n \tag{3}$$

where

$$r_i = \frac{\sqrt{p_i l_i}}{\sum_{j=1}^{n} \sqrt{p_j l_j}}, i = 1 \text{ to } n. \tag{4}$$

Here, p_i is the access probability of item i, l_i is the length of item i in bytes and n is the number of items chosen for broadcast.

Step (iii). The Secondary Hash Function (SHF) given by equations (5) and (6) is applied to determine the position of each item in their respective channels.

$$H(k,1) = [(A * k + B) \mod 2^{31}] \,\%L \tag{5}$$

where $A = 1103515245$, $B = 12345$ and $L =$ Length of broadcast cycle

$$H(k,n) = \left(H(k,1) + \frac{2n - 2^{\lceil \log_2 n \rceil} - 1}{2^{\lceil \log_2 n \rceil}} L\right) \mod L \tag{6}$$

where, $H(k,1)$ gives the position of the first instance of the data item with key k, and $H(k,n)$, with $n = 2$ to M, where M is the replication bound for an item, which gives the position of the remaining instances in an almost equally spaced manner.

Step (iv). Collisions may occur and holes may be present in the schedule constructed so far. These are resolved by a simple mechanism in which chaining is used to hash the colliding items and distance pointers are maintained to access chained items.

Step (v). The items thus identified are filled into "slots" and are broadcasted.

3.2. Updating of Access Probabilities

Data items are replicated based on their popularity and broadcasted. Access probabilities are continually maintained to estimate this popularity. Mobile clients are required to send responses to each item they access in the broadcast. This re-calculates access probabilities of data items after the k^{th} cycle in the broadcast schedule based on these responses and updates the Master Table maintained in the server.

The algorithm is as follows:

i. Maintain an access counter for each data item that clients access or make an explicit request for.

ii. Calculate the Normalized Response for an item i, $\beta(i)$ which represents the normalized environmental response after the server's k^{th} broadcast,
$\beta(i)$ = Ratio between the number of accesses of items i and the number of clients in the BS.

iii. Compute $P_i(k+1)$ which is the Update Access Probability of item i after the k^{th} cycle using equation (7),

$$p_i(k+1) = p_i(k) + L(1-\beta(k)) \sum_{j \neq i} (p_j(k)) \tag{7}$$

iv. Obtain $P_j(k+1)$ which is the Update Access Probability of all other items calculated using the formula,

$$p_j(k+1) = p_j(k) - L(1-\beta(k))(p_j(k)), \forall j \neq i \tag{8}$$

where L is the convergence factor that reflects the speed versus accuracy. In this system, this factor is 0.15.

3.3. Profile Management

A client initially submits its profile (interests) to the home Base Station. The submitted user profiles are analyzed to identify the common interests of the clients in that cell. Obviously, data items that satisfy most clients should be more readily available. Access probabilities of these items are updated to reflect the commonalities in interests. Access probabilities affect the frequency of broadcasts of the item; hence popular items are comparatively more easily available. When a client moves to some other location, the destination base station gets its profile from the old base station and uses it to update its access probabilities.

3.4. Efficient Client Access Algorithm

This algorithm is designed for accessing the interested data items by the mobile clients using the hashing scheme.

i. Let k be the key of the requested data item. Tune into some channel C_i to obtain primary hash function (PHF).

ii. Apply PHF to determine channel C_j in which this data item is being broadcasted.

iii. If $i \neq j$, hop to channel C_j and obtain the secondary hash function (SHF).

iv. Calculate the M slots using SHF which are the potential locations of the requested data item.

v. Sort the slot numbers in increasing order of their distances ahead of the initial probing slot and store them in a queue.

vi. Search the slots in this queue for the data item following the distance pointers if needed.

vii. If the data item is found, send a response pulse to the base station and cache the data item.

viii. Corrupt data items are identified by their CRC32 checksums and if a data item is corrupt, access its next instance.

ix. If the queue becomes empty, then the requested data item is not in broadcast. Hence send an *explicit pull request* to the server through a dedicated channel, wait for the server to respond and cache the data that arrived from the server.

3.5. Cache Management

A cache entry contains the following information, Key, Data – the real information, Valid – a flag indicating the validity of the entry, MobilityEffect – a flag indicating whether the cached item is mobility dependent, AVI – the expected validity interval of the item after which the item may be expected, LastUpdateTime – time of last update in the database, LastAccessTime – time of last access by the mobile client, NumberofHits – number of cache hits so far. Cache management includes:

Cache Replacement

The basic factors of cache replacement such as LastAccessTime and NumberofHits are utilized in this system. An invalid entry that has the minimum last access time is chosen as the victim for cache replacement. If all the entries are valid, the one with the least number of hits (LFU) is chosen to be the victim.

Cache Invalidation due to Mobility

In this system, cache invalidation is done by using the location of the mobile user and ***Invalidation by Absolute Validity Interval*** scheme. When a client moves to a new location, some data items may need to be invalidated. This is achieved by having a *Mobility Effect* bit in the cache, which is set for such data items.

Invalidation of cache entries is also done using Invalidation by Absolute Validity Interval scheme [4]. Both the server and client maintain a running estimate of update intervals for each data item termed AVI. The client stores the AVI for the data item in the cache and invalidates that entry when AVI lapses. This is the case when the estimate AVI increases at server. When the estimate AVI decreases at server, it sends an ***Invalidation Report*** (IR) that contains the key of data item and the new AVI.

Formally, the items that satisfy the equation 9 proposed in [4] are added to the IR,

$$T_{\text{update}(i,n)} - T_{\text{update}(i,n-1)} < \text{AVI}(i) \times (1 - F_i) \quad (9)$$

where $T_{\text{update}(i,n)}$ is the timestamp of n^{th} update on data item i; AVI(i) is the AVI of data item i; and F_i is the AVI tolerance for data item i.

4. System Architecture

There are three core entities in the proposed model – the Mobile Switching Centre (MSC), the Base Station (BS) and the mobile client. MSC serves as an interface between base stations and enables communication between them. Base station takes care of gathering data that may be of interest to clients in its cell and broadcasts the data in repetitive cycles. Mobile clients, when given a query, access the broadcast for retrieving the data.

4.1. Server (Base Station)

Each server maintains its own database that contains a Master Table to store the common attributes of all data items and a Profile Table to store the profiles of the mobile clients currently located under that base station. The schema of the master table contains Key – a 32 bit integer, Access Probability – the probability that the data item will be accessed, Number of pull requests – the number of explicit requests during the current broadcast cycle, Time of Last Update – the time at which the item was last updated, Absolute Validity Interval – the expected time duration after which the item may be updated, Pointer to Data Item – a string indicating the table name and record number of the data item.

Master Table Manager is a component that initializes the Master Table by calculating keys, access probabilities of all items in the server database. The initial access probabilities are calculated from the profiles submitted by the users when they register in a base station.

The real time characteristic of certain data items cause frequent updates in the database. Database Manager handles these updates to the database. When a new entry arrives, the DB manager calculates its key, calculates new Absolute Validity Interval (AVI), based on the old AVI and last update time, and updates the Master Table. If the data item were updated sooner than expected, there would be a decrease in its AVI. To alert clients that might have cached the old value of the data item, this component broadcasts an invalidation report (IR) which is of the following format: Key – the key of the updated data item, Last Update Time – the time of the update, Absolute Validity Interval – the new expected interval after which the item may get updated.

Mobile clients are required to submit their interests to the base station when registering them. Profile Manager reads the profiles submitted by the clients and loads them into the server database. Clients can update their profiles at any time and this component also handles these profile updates. The profile is stored along with the mobile ID in the Profile Table in the database.

Access Probability Manager maintains the access probability for each data item initially based on the profile and then with the user responses to access in the broadcast. Broadcast Cycle Constructor constructs the broadcast cycle based on the previously mentioned steps and broadcasts to mobile clients. Mobile clients can send explicit pull requests to the server in case their request could not be satisfied from the broadcast data. Pull Manager handles these explicit requests. It searches the server's database for

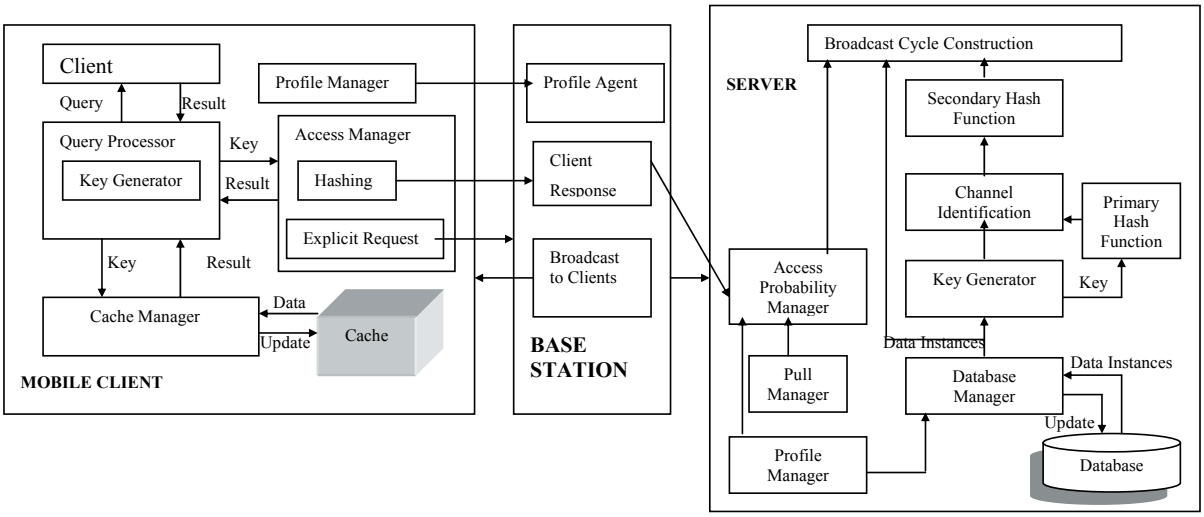

Figure 1. System architecture.

the requested item and returns it to the client. The user is informed if the requested item is not found in the database.

4.2. Mobile Client

On receiving a query from the user, the Query Processor computes the key of the requested item. It then searches the cache using the key and returns the item in case of a cache hit. In case of a cache miss, this requests the Access Manager to listen to the broadcast and retrieve the item. Access Manager is the one that tunes into the broadcast to listen to the information that is disseminated periodically by the server. Cache Manager is important in the mobile since it leads to an improvement in time and power spent by the mobile client. The Cache Manager searches the cache when a client's request arrives. If the requested item is found and is valid (i.e. its last update time + AVI >= current time), updates the time of last access, increments the hit count and returns the item. When a new item is to be inserted into the cache, the Cache Manager searches for an Empty entry and inserts the data item if one is available. Otherwise, cache replacement occurs. An Invalid entry that has the minimum last access time is chosen as the victim. If all the entries are Valid, the one with the least number of hits (LFU) is chosen to be the victim. The Cache Manager invalidates items on three counts: A cache item is invalidated if its AVI elapses (current time > last update time + AVI). An item is invalidated if it is mentioned in an Invalidation Report broadcast by the server and read by the Cache Manager. And, finally, the mobility of clients may cause some items to be invalidated, for instance, information about hospitals.

Profile Manager in the mobile client allows a new user to register with the base station by submitting its interests. It also allows existing users to update their already submitted profiles. This is entrusted with the lone task of sending the profile to the base station.

Mobility of the client is also taken care of in this system. When a client wants to move to a different cell, this module sends a request notification to the current BS. When the request is granted, the Mobility Manager updates the existing base station information like which are the channels to be listened etc. The mobile client can move to the new cell and resume its activities after this updating.

5. Simulation and Results

This section explains the simulation performed and the results obtained thereof. To simulate the evaluation of the proposed hashing scheme, first the framework is developed. The framework is modeled using the JSim Simulator. All necessary components, of mobile environment (viz Mobile Unit – which presents the query,

Base Station Unit and Mobile Switching Centre) required for this simulation are modeled as JSim Processes.

This section also evaluates the performance of the proposed broadcast scheme. Performance is measured in terms of the two parameters, access latency and tuning time. This scheme was tested under diffferent scenarios by varying channel sizes, applying hashing, client-side caching; and performance comparisons were done. Figure 2 shows the performance of the proposed scheme against a broadcasting scheme that uses no auxiliary information. Figure 3 and Figure 4 illustrate the tuning time and access latency of the proposed scheme when the number of channels was kept as 0, 5 and 10 and the number of cache entries was kept as 5, 10 and 15. Figure 5 shows the energy consumption in a typical mobile system.

5.1. Performance Comparison of the Hashing Scheme with Varying Number of Channels

From Figure 2, it is observed that the tuning time decreases considerably with hashing. It is because when hashing is used with non flat data broadcasts which are replicated based on user profiles and access probabilities, the mobile unit can find out the slot number in which the required data will arrive indicating the data arrival time, and thus it calculates the delay after which the slot will get broadcasted. Hence, the time it stays in active mode is reduced and thus considerable power in the mobile unit is saved.

But, without hashing or any other indexing scheme, a client has to sequentially search the broadcast information to find the data it needs. It has to tune the brodcast channel and stay in the active mode until it finds the required data item. Hence, there is an increase in tuning time as shown in Figure 2.

When the number of channels available for broadcast increases, there is a decrease in tuning time of the mobile client. An increase in the number of channels directly implies that there are more data items available for access at a given time. Data items can be replicated and broadcasted more frequently, which enables quicker access.

5.2. Performance Comparison with and without Cache of the Proposed Scheme with Varying Number of Channels

It is observed from Figure 3 that an increase in the number of channels and size of the cache causes the access latency to decrease. When there is no cache, a client has to access the broadcast information each time to service each query. However, with the cache, average access latency is reduced. Only when the item is not found in the cache, the client accesses the broadcast information, introducing latency. As the number of cache entries increases, more items can be fetched quickly and cache replacement occurs less often and all this leads to a decrease in average access latency.

An increase in the number of channels causes a decrease in access latency. The replication bound of each item increases as the number of channel increases. If a mobile client misses to access an item from the broadcast, it can read the next instance of the same item found in another channel of the same cycle.

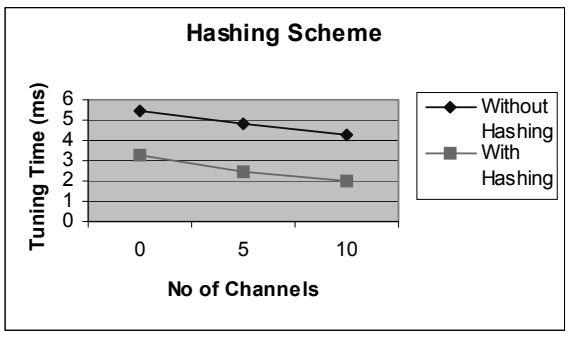

Figure 2. Tuning time reductions with hashing.

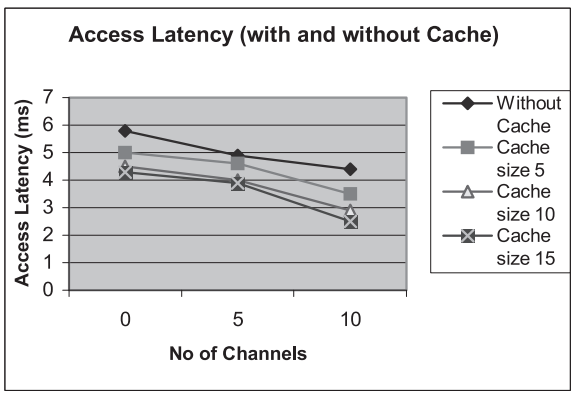

Figure 3. Access latency with caches of different sizes.

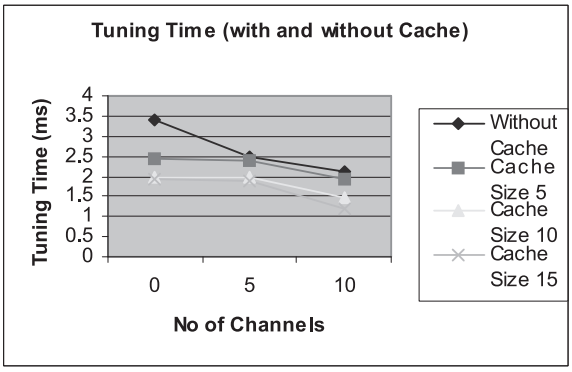

Figure 4. Tuning time with caches of different sizes.

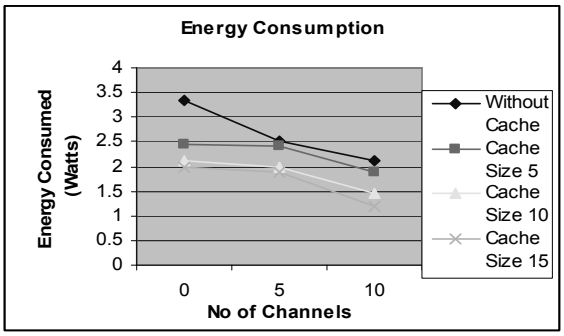

Figure 5. Energy consumption graph.

From Figure 4, it is deduced that with an increase in the number of channels and cache size, tuning time is reduced for the mobile unit. Without a cache, the client tunes into broadcast for each item and hence there is some amount of tuning time for each query serviced. If a cached item is queried the overall tuning time reduces. The proposed scheme reduces both access latency and tuning time concurrently. Popular items are replicated to reduce access latency and this causes tuning time for these items to decrease as well. This feature helps in achieving less tuning time when the access latency is reduced.

5.3. Energy Consumption in the Proposed Scheme – A Comparison with Cache and with Varying Number of Channels

The energy consumed by a typical mobile system is depicted in Figure 5. The system is assumed to spend 0.95 watts in active mode and 0.06 watts in the power-saving doze mode.

$$E = (\text{accesslatency} - \text{tuningtime}).r_{\text{doze}} + \text{tuningtime}.r_{\text{active}} \quad (10)$$

The energy consumption E by a mobile system is directly proportional to the access latency and tuning time given by the equation (10). When caching is used and broadcast is done on multiple channels, both of these parameters are decreased, which leads to a less consumption of energy in the mobile unit.

6. Conclusion and Future Work

Wireless data broadcast that allows simultaneous access by an arbitrary number of clients is a very efficient and scalable method of information dissemination. This paper proposes an efficient hashing scheme for broadcasting non-flat broadcast data over multiple channels. Moreover, specific characteristics of the broadcast environment are discussed in addition to the study of several existing single-channel and multi-channel broadcast schemes. It is found that each of them has its own disadvantages, like indexing schemes increased tuning time and so on. The proposed scheme caters to multi-channel broadcast, while eliminating the disadvantages of the previous schemes. While existing schemes reduce either access latency or tuning time, the proposed scheme reduces both in an integrated manner.

The major contributions and advantages of this paper are as follows.

- A Hashing scheme for multi-channel broadcasting is proposed which optimizes both access latency and tuning time in an integrated fashion.
- Data items are replicated based on their access probabilities and user profiles which further decreases access latency. The popularity of data items is continuously measured and the broadcast is streamlined to suit the client needs.
- Error resilience is also achieved by replicas of data items.
- The client access mechanism proposed in this paper is simple and fast.

- Caching is introduced for enhancing the performance of the system.

The results obtained from this scheme show that the performance of the scheme is better because of many reasons such as multiple channels, shorter broadcast cycles and caching at mobile clients. However, this system does not consider the processing of window queries like, "Find a restaurant within 200 m" using hash functions, as they cannot aid in resolving the Cartesian coordinates. We are investigating methods to allow such queries using our hashing scheme as a future work. Also, the scheme can be extended to free mobile clients from mandatorily sending response pulses to data items that they access from the broadcast information.

References

[1] ANDRE SEIFERT, JEN-JOU HUNG, FlexSched: A Flexible Data Schedule Generator for Multi-channel Broadcast Systems, University of Konstanz, Technical Report, 2005.

[2] D. BARBARA AND T. IMIELINSKI, Sleepers and Workaholics: Caching Strategies in Mobile Environments. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, (1994) pp. 1–12.

[3] JIANLIANG XU, WANG-CHIEN LEE, XUEYAN TANG, QING GAO, AND SHANPING LI, An Error-resilient and Tunable Distributed Indexing Scheme for Wireless Data Broadcast. *IEEE Transactions on Knowledge and Data Engineering*, (2006) **18**(3), pp. 392–403.

[4] JOE CHUN-HUNG YUEN, EDWARD CHAN, KAM-YIU LAM AND H. W. LEUNG, Cache Invalidation Scheme for Mobile Computing Systems with Real-time Data. *SIGMOD Record* (2000) **29**(4), pp. 34–39.

[5] KONSTANTINOS STATHATOS, NICK ROUSSOPOULOS, JOHN S. BARAS, Adaptive Data Broadcast in Hybrid Networks. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, (1997) pp. 326–335.

[6] NITIN VAIDYA, SOHAIL HAMEED, Scheduling Data Broadcast in Asymmetric Communication Environments. In *Proceedings of the 20th International Conference on Data Engineering*, (1999) pp. 171–182.

[7] PETROS NICOPOLITIDIS, GEORGIOS I. PAPADIMITRIOU AND ANDREAS S. POMPORTSIS, Using Learning Automata for Adaptive Push-based Data Broadcasting in Asymmetric Wireless Environments. *IEEE Transactions on Vehicular Technology*, (2002) **51**(6), pp. 1652–1660.

[8] SHIJUN DDYU, STEFANO SPACCAPIETRA, NADINE CULLOT, MARIE-AUDE UFAURE, User Profiles in Location-based Services: Make Humans More Nomadic and Personalized. In the *Proceedings of IASTED International Conference on Databases and Applications*, (2004) pp. 25–30.

[9] SILVANA BADALONI, MASSIMILIANO GIACOMIN, A Fuzzy Extension of Allen's Interval Algebra. *Proceedings of the 6th Congress of the Italian Association for Artificial Intelligence on Advances in Artificial Intelligence*, (1999) **1792**, pp. 155–165.

[10] T. IMIELINSKI, S. VISWANATHAN, AND B. R. BADRINATH, Power Efficient Filtering of Data on Air. *Proceedings of the Fourth International Conference Extending Database Technology*, (1994) pp. 245–258.

[11] T. IMIELINSKI, S. VISWANATHAN, AND B. R. BADRINATH, Data on Air: Organization and Access. *IEEE Transactions on Knowledge and Data Engineering*, (1997) **9**(3), pp. 353–372.

[12] W. G. YEE AND S. B. NAVATHE, Efficient Data Access to Multi-channel Broadcast Programs. *ACM Conference on Knowledge Management*, (2003) pp. 153–160.

[13] YUXIA YAO, XUEYAN TANG, EE-PENG LIM, AIXIN SUN, An Energy – Efficient and Access Latency Optimized Indexing Scheme for Wireless Data Broadcast. *IEEE Transactions on Knowledge and Data Engineering*, (2006) **18**(8), pp. 1111–1124.

Received: August, 2007
Revised: January, 2008
Accepted: April, 2008

Contact address:
Muthuswamy Vijayalakshmi
Arputharaj Kannan
Dept of Computer Science and Engineering
College of Engineering
Anna University
Chennai-25, India
e-mail: viji_mathan@yahoo.com
kannan@annauniv.edu

MUTHUSWAMY VIJAYALAKSHMI is a lecturer at the Department of CSE, College of Engineering, Guindy, Anna University, India. She has 6 years of teaching experience. Currently, she is working toward her PhD in computer science and engineering at Anna University, India. Her research areas include mobile computing, databases, and artificial intelligence.

ARPUTHARAJ KANNAN is a professor at the Department of CSE, College of Engineering, Guindy, Anna University, India. He has 18 years of teaching experience. He received his PhD in computer science and engineering from Anna University, India in 2001. His research areas include software engineering, database management systems, and artificial intelligence.