Location based Indexing Scheme for DAYS

Debopam Acharya and Vijay Kumar[1]
Computer Science and Informatics
University of Missouri-Kansas City
Kansas City, MO 64110
dargc(kumarv)@umkc.edu

ABSTRACT
Data dissemination through wireless channels for broadcasting information to consumers is becoming quite common. Many dissemination schemes have been proposed but most of them *push* data to wireless channels for general consumption. Push based broadcast [1] is essentially asymmetric, i.e., the volume of data being higher from the server to the users than from the users back to the server. Push based scheme requires some indexing which indicates when the data will be broadcast and its position in the broadcast. Access latency and tuning time are the two main parameters which may be used to evaluate an indexing scheme. Two of the important indexing schemes proposed earlier were tree based and the exponential indexing schemes. None of these schemes were able to address the requirements of location dependent data (LDD) which is highly desirable feature of data dissemination. In this paper, we discuss the broadcast of LDD in our project DAta in Your Space (DAYS), and propose a scheme for indexing LDD. We argue that this scheme, when applied to LDD, significantly improves performance in terms of tuning time over the above mentioned schemes. We prove our argument with the help of simulation results.

Categories and Subject Descriptors
H.3.1 [**Information Systems**]: Information Storage and Retrieval – *content analysis and indexing;* H.3.3 [**Information Systems**]: Information Storage and Retrieval – *information search and retrieval.*

General Terms
Algorithms, Performance, Experimentation

Keywords
Wireless data broadcast, indexing, location based services, data staging, location dependent data

1. INTRODUCTION
Wireless data dissemination is an economical and efficient way to make desired data available to a large number of mobile or static users. The mode of data transfer is essentially asymmetric, that is, the capacity of the transfer of data (downstream communication) from the server to the client (mobile user) is significantly larger than the client or mobile user to the server (upstream communication). The effectiveness of a data dissemination system is judged by its ability to provide user the required data at anywhere and at anytime. One of the best ways to accomplish this is through the dissemination of highly personalized Location Based Services (LBS) which allows users to access personalized location dependent data. An example would be someone using their mobile device to search for a vegetarian restaurant. The LBS application would interact with other location technology components or use the mobile user's input to determine the user's location and download the information about the restaurants in proximity to the user by tuning into the wireless channel which is disseminating LDD.

We see a limited deployment of LBS by some service providers. But there are every indications that with time some of the complex technical problems such as uniform location framework, calculating and tracking locations in all types of places, positioning in various environments, innovative location applications, etc., will be resolved and LBS will become a common facility and will help to improve market productivity and customer comfort. In our project called DAYS, we use wireless data broadcast mechanism to push LDD to users and mobile users monitor and tune the channel to find and download the required data. A simple broadcast, however, is likely to cause significant performance degradation in the energy constrained mobile devices and a common solution to this problem is the use of efficient air indexing. The indexing approach stores control information which tells the user about the data location in the broadcast and how and when he could access it. A mobile user, thus, has some free time to go into the doze mode which conserves valuable power. It also allows the user to personalize his own mobile device by selectively tuning to the information of his choice.

Access efficiency and energy conservation are the two issues which are significant for data broadcast systems. Access efficiency refers to the latency experienced when a request is initiated till the response is received. Energy conservation [7, 10] refers to the efficient use of the limited energy of the mobile device in accessing broadcast data. Two parameters that affect these are the *tuning time* and the *access latency*. Tuning time refers to the time during which the mobile unit (MU) remains in active state to tune the channel and download its required data. It can also be defined as the number of buckets tuned by the mobile device in active state to get its required data. Access latency may be defined as the time elapsed since a request has been issued till the response has been received.

[1]This research was supported by a grant from NSF IIS-0209170.

MobiDE'05, June 12, 2005, Baltimore, Maryland, USA.

Several indexing schemes have been proposed in the past and the prominent among them are the tree based and the exponential indexing schemes [17]. The main disadvantages of the tree based schemes are that they are based on centralized tree structures. To start a search, the MU has to wait until it reaches the root of the next broadcast tree. This significantly affects the tuning time of the mobile unit. The exponential schemes facilitate index replication by sharing links in different search trees. For broadcasts with large number of pages, the exponential scheme has been shown to perform similarly as the tree based schemes in terms of access latency. Also, the average length of broadcast increases due to the index replication and this may cause significant increase in the access latency. None of the above indexing schemes is equally effective in broadcasting location dependent data. In addition to providing low latency, they lack properties which are used to address LDD issues. We propose an indexing scheme in DAYS which takes care of some these problems. We show with simulation results that our scheme outperforms some of the earlier indexing schemes for broadcasting LDD in terms of tuning time.

The rest of the paper is presented as follows. In section 2, we discuss previous work related to indexing of broadcast data. Section 3 describes our DAYS architecture. Location dependent data, its generation and subsequent broadcast is presented in section 4. Section 5 discusses our indexing scheme in detail. Simulation of our scheme and its performance evaluation is presented in section 6. Section 7 concludes the paper and mentions future related work.

2. PREVIOUS WORK

Several disk-based indexing techniques have been used for air indexing. Imielinski et al. [5, 6] applied the B+ index tree, where the leaf nodes store the arrival times of the data items. The distributed indexing method was proposed to efficiently replicate and distribute the index tree in a broadcast. Specifically, the index tree is divided into a replicated part and a non replicated part. Each broadcast consists of the replicated part and the non-replicated part that indexes the data items immediately following it. As such, each node in the non-replicated part appears only once in a broadcast and, hence, reduces the replication cost and access latency while achieving a good tuning time. Chen et al. [2] and Shivakumar et al. [8] considered unbalanced tree structures to optimize energy consumption for non-uniform data access. These structures minimize the average index search cost by reducing the number of index searches for hot data at the expense of spending more on cold data. Tan and Yu discussed data and index organization under skewed broadcast Hashing and signature methods have also been suggested for wireless broadcast that supports equality queries [9]. A flexible indexing method was proposed in [5]. The flexible index first sorts the data items in ascending (or descending) order of the search key values and then divides them into p segments. The first bucket in each data segment contains a control index, which is a binary index mapping a given key value to the segment containing that key, and a local index, which is an m-entry index mapping a given key value to the buckets within the current segment. By tuning the parameters of p and m, mobile clients can achieve either a good tuning time or good access latency. Another indexing technique proposed is the exponential indexing scheme [17]. In this scheme, a parameterized index, called the exponential index is used to optimize the access latency or the tuning time. It facilitates index replication by linking different search trees. All of the above mentioned schemes have been applied to data which are non related to each other. These non related data may be clustered or non clustered. However, none of them has specifically addressed the requirements of LDD. Location dependent data are data which are associated with a location. Presently there are several applications that deal with LDD [13, 16]. Almost all of them depict LDD with the help of hierarchical structures [3, 4]. This is based on the containment property of location dependent data. The Containment property helps determining relative position of an object by defining or identifying locations that contains those objects. The subordinate locations are hierarchically related to each other. Thus, Containment property limits the range of availability or operation of a service. We use this containment property in our indexing scheme to index LDD.

3. DAYS ARCHITECTURE

DAYS has been conceptualized to disseminate topical and non-topical data to users in a local broadcast space and to accept queries from individual users globally. Topical data, for example, weather information, traffic information, stock information, etc., constantly changes over time. Non topical data such as hotel, restaurant, real estate prices, etc., do not change so often. Thus, we envision the presence of two types of data distribution: In the first case, server pushes data to local users through wireless channels. The other case deals with the server sending results of user queries through downlink wireless channels. Technically, we see the presence of two types of queues in the pull based data access. One is a heavily loaded queue containing globally uploaded queries. The other is a comparatively lightly loaded queue consisting of locally uploaded queries. The DAYS architecture [12] as shown in figure 1 consists of a Data Server, Broadcast Scheduler, DAYS Coordinator, Network of LEO satellites for global data delivery and a Local broadcast space. Data is pushed into the local broadcast space so that users may tune into the wireless channels to access the data. The local broadcast space consists of a broadcast tower, mobile units and a network of data staging machines called the surrogates. Data staging in surrogates has been earlier investigated as a successful technique [12, 15] to cache users' related data. We believe that data staging can be used to drastically reduce the latency time for both the local broadcast data as well as global responses. Query request in the surrogates may subsequently be used to generate the popularity patterns which ultimately decide the broadcast schedule [12].

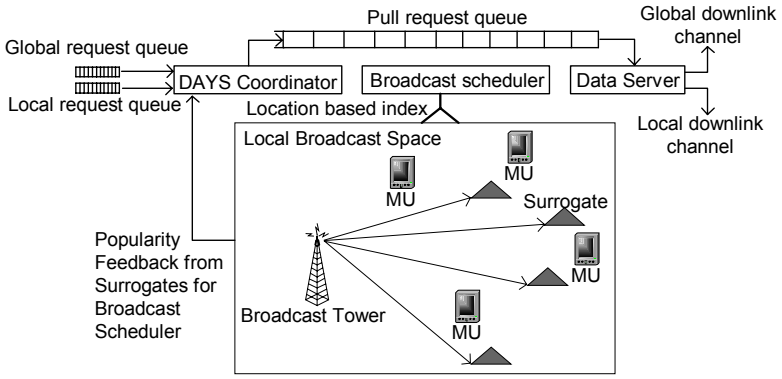

Figure 1. DAYS Architecture

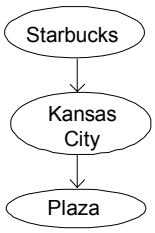

Figure 2. Location Structure of Starbucks, Plaza

4. LOCATION DEPENDENT DATA (LDD)

We argue that incorporating location information in wireless data broadcast can significantly decrease the access latency. This property becomes highly useful for mobile unit which has limited storage and processing capability. There are a variety of applications to obtain information about traffic, restaurant and hotel booking, fast food, gas stations, post office, grocery stores, etc. If these applications are coupled with location information, then the search will be fast and highly cost effective. An important property of the locations is *Containment* which helps to determine the relative location of an object with respect to its parent that contains the object. Thus, Containment limits the range of availability of a data. We use this property in our indexing scheme. The database contains the broadcast contents which are converted into LDD [14] by associating them with respective locations so that it can be broadcasted in a clustered manner. The clustering of LDD helps the user to locate information efficiently and supports containment property. We present an example to justify our proposition.

Example: *Suppose a user issues query "Starbucks Coffee in Plaza please." to access information about the Plaza branch of Starbucks Coffee in Kansas City. In the case of location independent set up the system will list all Starbucks coffee shops in Kansas City area. It is obvious that such responses will increase access latency and are not desirable. These can be managed efficiently if the server has location dependent data, i.e., a mapping between a Starbucks coffee shop data and its physical location. Also, for a query including range of locations of Starbucks, a single query requesting locations for the entire region of Kansas City, as shown in Figure 2, will suffice. This will save enormous amount of bandwidth by decreasing the number of messages and at the same time will be helpful in preventing the scalability bottleneck in highly populated area.*

4.1 Mapping Function for LDD

The example justifies the need for a mapping function to process location dependent queries. This will be especially important for pull based queries across the globe for which the reply could be composed for different parts of the world. The mapping function is necessary to construct the broadcast schedule.

We define Global Property Set (GPS) [11], Information Content (IC) set, and Location Hierarchy (LH) set where IC $\subseteq$ GPS and LH $\subseteq$ GPS to develop a mapping function. LH = $\{l_1, l_2, l_3 \ldots, l_k\}$ where l_i represent locations in the location tree and IC = $\{ic_1, ic_2, ic_3, \ldots, ic_n\}$ where ic_i represent information type. For example, if we have traffic, weather, and stock information are in broadcast then IC = $\{ic_{traffic}, ic_{weather}$, and $ic_{stock}\}$. The mapping scheme must be able to identify and select an IC member and a LH node for (a) correct association, (b) granularity match, (c) and termination condition. For example, *weather* $\in$ *IC* could be associated with a *country* or a *state* or a *city* or a *town* of LH. The granularity match between the weather and a LH node is as per user requirement. Thus, with a coarse granularity *weather* information is associated with a *country* to get country's *weather* and with *town* in a finer granularity. If a *town* is the finest granularity, then it defines the terminal condition for association between IC and LH for weather. This means that a user cannot get weather information about subdivision of a *town*. In reality weather of a subdivision does not make any sense.

We develop a simple heuristic mapping approach scheme based on user requirement. Let $IC = \{m_1, m_2, m_3, \ldots, m_k\}$, where m_i represent its element and let $LH = \{n_1, n_2, n_3, \ldots, n_l\}$, where n_i represents LH's member. We define GPS for *IC* (GPS_{IC}) $\subseteq$ GPS and for *LH* (GPS_{LH}) $\subseteq$ GPS as $GPS_{IC} = \{P_1, P_2, \ldots, P_n\}$, where $P_1, P_2, P_3, \ldots, P_n$ are properties of its members and $GPS_{LH} = \{Q_1, Q_2, \ldots, Q_m\}$ where $Q_1, Q_2, \ldots, Q_m$ are properties of its members.

The properties of a particular member of *IC* are a subset of GPS_{IC}. It is generally true that (*property* set ($m_i \in IC$) $\cup$ *property set* ($m_j \in IC$)) $\neq \emptyset$, however, there may be cases where the intersection is not null. For example, stock $\in$ *IC* and movie $\in$ *IC* rating do not have any property in common. We assume that any two or more members of *IC* have at least one common geographical property (i.e. location) because DAYS broadcasts information about those categories, which are closely tied with a location. For example, *stock* of a company is related to a *country*, *weather* is related to a *city* or *state*, etc.

We define the property subset of $m_i \in IC$ as $PSm_i \; \forall \, m_i \in IC$ and $PSm_i = \{P_1, P_2, \ldots, P_r\}$ where $r \leq n$. $\forall P_r \{P_r \in PSm_i \rightarrow P_r \in GPS_{IC}\}$ which implies that $\forall i, PSm_i \subseteq GPS_{IC}$. The geographical properties of this set are indicative of whether $m_i \in IC$ can be mapped to only a single granularity level (i.e. a single location) in *LH* or a multiple granularity levels (i.e. more than one nodes in

the hierarchy) in *LH*. How many and which granularity levels should a m_i map to, depends upon the level at which the service provider wants to provide information about the m_i in question. Similarly we define a property subset of *LH* members as $PSn_j \forall\ n_j \in LH$ which can be written as $PSn_j = \{Q_1, Q_2, Q_3, \ldots, Q_s\}$ where $s \leq m$. In addition, $\forall Q_s \{Q_s \in PSn_j \rightarrow Q_s \in GPS_{LH}\}$ which implies that $\forall j, PSn_j \subseteq GPS_{LH}$.

The process of mapping from *IC* to *LH* is then identifying for some $m_x \in IC$ one or more $n_y \in LH$ such that $PSm_x \cap PSn_v \neq \phi$. This means that when m_x maps to n_y and all children of n_y if m_x can map to multiple granularity levels or m_x maps only to n_y if m_x can map to a single granularity level.

We assume that new members can join and old member can leave IC or LH any time. The deletion of members from the IC space is simple but addition of members to the IC space is more restrictive. If we want to add a new member to the IC space, then we first define a property set for the new member: $PSm_{new\_m} = \{P_1, P_2, P_3, \ldots, P_t\}$ and add it to the IC only if the condition: $\forall P_w \{P_w \in PSp_{new\_m} \rightarrow P_w \in GPS_{IC}\}$ is satisfied. This scheme has an additional benefit of allowing the information service providers to have a control over what kind of information they wish to provide to the users. We present the following example to illustrate the mapping concept.

IC = {Traffic, Stock, Restaurant, Weather, Important history dates, Road conditions}

LH = {Country, State, City, Zip-code, Major-roads}

GPS_{IC} *= {Surface-mobility, Roads, High, Low, Italian-food, StateName, Temp, CityName, Seat-availability, Zip, Traffic-jams, Stock-price, CountryName, MajorRoadName, Wars, Discoveries, World}*

GPS_{LH} *= {Country, CountrySize, StateName, CityName, Zip, MajorRoadName}*

$Ps(IC_{Stock})$ *= {Stock-price, CountryName, High, Low}*

$Ps(IC_{Traffic})$ *= {Surface-mobility, Roads, High, Low, Traffic-jams, CityName}*

$Ps(IC_{Important\ dates\ in\ history})$ *= {World, Wars, Discoveries}*

$Ps(IC_{Road\ conditions})$ *= {Precipitation, StateName, CityName}*

$Ps(IC_{Restaurant})$ *= {Italian-food, Zip code}*

$Ps(IC_{Weather})$ *= {StateName, CityName, Precipitation, Temperature}*

$PS(LH_{Country})$ *= {CountryName, CountrySize}*

$PS(LH_{State}$ *= {StateName, State size},*

$PS(LH_{City})$ *={CityName, City size}*

$PS(LH_{Zipcode})$ *= {ZipCodeNum }*

$PS(LH_{Major\ roads})$ *= {MajorRoadName}*

Now, only $PS(IC_{Stock}) \cap PS_{Country} \neq \phi$. In addition, $PS(IC_{Stock})$ indicated that *Stock* can map to only a single location *Country*. When we consider the member *Traffic* of *IC* space, only $PS(IC_{Traffic}) \cap PS_{city} \neq \phi$. *As* $PS(IC_{Traffic})$ indicates that *Traffic* can map to only a single location, it maps only to *City* and none of its children. Now unlike *Stock,* mapping of *Traffic* with *Major roads,* which is a child of *City,* is meaningful. However service providers have right to control the granularity levels at which they want to provide information about a member of *IC* space.

$PS(IC_{Road\ conditions}) \cap PS_{State} \neq \phi$ and $PS(IC_{Road\ conditions}) \cap PS_{City} \neq \phi$. So *Road conditions* maps to *State* as well as *City*. As $PS(IC_{Road\ conditions})$ indicates that *Road conditions* can map to multiple granularity levels, *Road conditions* will also map to *Zip Code* and *Major roads*, which are the children of *State* and *City*. Similarly, *Restaurant* maps only to *Zip code*, and *Weather* maps to *State*, *City* and their children, *Major Roads* and *Zip Code.*

5. LOCATION BASED INDEXING SCHEME

This section discusses our location based indexing scheme (LBIS). The scheme is designed to conform to the LDD broadcast in our project DAYS. As discussed earlier, we use the containment property of LDD in the indexing scheme. This significantly limits the search of our required data to a particular portion of broadcast. Thus, we argue that the scheme provides bounded tuning time.

We describe the architecture of our indexing scheme. Our scheme contains separate data buckets and index buckets. The index buckets are of two types. The first type is called the Major index. The Major index provides information about the types of data broadcasted. For example, if we intend to broadcast information like Entertainment, Weather, Traffic etc., then the major index points to either these major types of information and/or their main subtypes of information, the number of main subtypes varying from one information to another. This strictly limits number of accesses to a Major index. The Major index never points to the original data. It points to the sub indexes called the Minor index. The minor indexes are the indexes which actually points to the original data. We called these minor index pointers as *Location Pointers* as they points to the data which are associated with a location. Thus, our search for a data includes accessing of a major index and some minor indexes, the number of minor index varying depending on the type of information.

Thus, our indexing scheme takes into account the hierarchical nature of the LDD, the Containment property, and requires our broadcast schedule to be clustered based on data type and location. The structure of the location hierarchy requires the use of different types of index at different levels. The structure and positions of index strictly depend on the location hierarchy as described in our mapping scheme earlier. We illustrate the implementation of our scheme with an example. The rules for framing the index are mentioned subsequently.

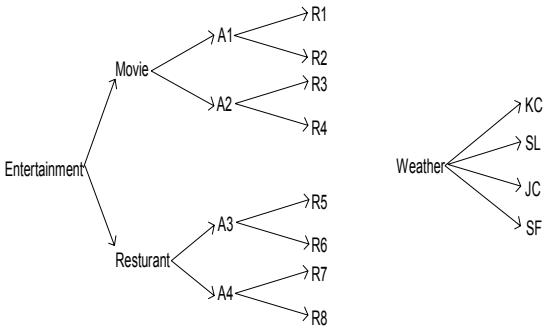

Figure 3. Location Mapped Information for Broadcast

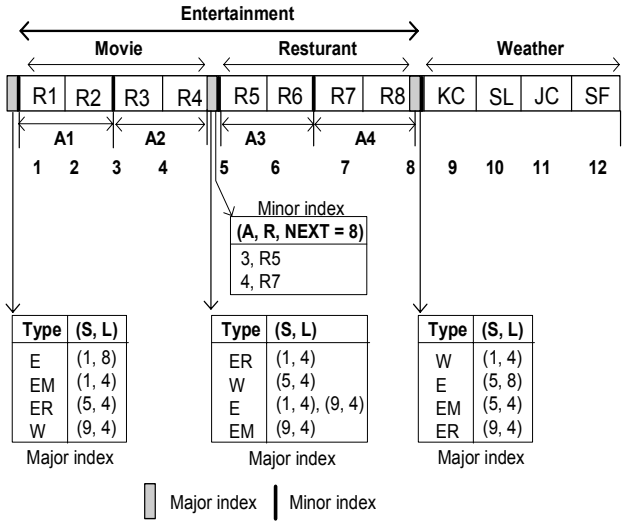

Figure 4. Data coupled with Location based Index

Example: Let us suppose that our broadcast content contains $IC_{entertainment}$ and $IC_{weather}$ which is represented as shown in Fig. 3. Ai represents Areas of City and Ri represents roads in a certain area. The leaves of *Weather* structure represent four cities. The index structure is given in Fig. 4 which shows the position of major and minor index and data in the broadcast schedule.

We propose the following rules for the creation of the air indexed broadcast schedule:

- The major index and the minor index are created.
- The major index contains the position and range of different types of data items (Weather and Entertainment, Figure 3) and their categories. The sub categories of Entertainment, Movie and Restaurant, are also in the index. Thus, the major index contains Entertainment (E), Entertainment-Movie (EM), Entertainment-Restaurant (ER), and Weather (W). The tuple (S, L) represents the starting position (S) of the data item and L represents the range of the item in terms of number of data buckets.
- The minor index contains the variables A, R and a pointer Next. In our example (Figure 3), road R represents the first node of area A. The minor index is used to point to actual data buckets present at the lowest levels of the hierarchy. In contrast, the major index points to a broader range of locations and so it contains information about main and sub categories of data.
- Index information is not incorporated in the data buckets. Index buckets are separate containing only the control information.
- The number of major index buckets *m=#(IC),* IC = {ic_1, ic_2, ic_3,…,ic_n} where ic_i represent information type and # represents the cardinality of the Information Content set IC. In this example, IC= {ic_{Movie}, $ic_{Weather}$, $ic_{Restaurant}$} and so #(IC) =3. Hence, the number of major index buckets is 3.
- Mechanism to resolve the query is present in the java based coordinator in MU. For example, if a query Q is presented as Q (Entertainment, Movie, Road_1), then the resultant search will be for the EM information in the major index. We say, *Q→EM*.

Our proposed index works as follows: Let us suppose that an MU issues a query which is represented by Java Coordinator present in the MU as "Restaurant information on Road 7". This is resolved by the coordinator as *Q→ER*. This means one has to search for ER unit of index in the major index. Let us suppose that the MU logs into the channel at R2. The first index it receives is a minor index after R2. In this index, value of Next variable = 4, which means that the next major index is present after bucket 4. The MU may go into doze mode. It becomes active after bucket 4 and receives the major index. It searches for ER information which is the first entry in this index. It is now certain that the MU will get the position of the data bucket in the adjoining minor index. The second unit in the minor index depicts the position of the required data R7. It tells that the data bucket is the first bucket in Area 4. The MU goes into doze mode again and becomes active after bucket 6. It gets the required data in the next bucket. We present the algorithm for searching the location based Index.

Algorithm 1 Location based Index Search in DAYS

1. Scan broadcast for the next index bucket, *found*=false
2. While (not *found*) **do**
3. if bucket is Major Index **then**
4. Find the *Type* & Tuple *(S, L)*
5. **if** *S* is greater than 1, go into doze mode for *S* seconds
6. **end if**
7. Wake up at the S^{th} bucket and observe the Minor Index
8. end if
9. if bucket is Minor Index **then**
10. if $Type_{Requested}$ not equal to $Type_{found}$ and $(A,R)_{Request}$ not equal to $(A,R)_{found}$ **then**
11. Go into doze mode till *NEXT* & repeat from step 3
12. end if
13. **else** find entry in Minor Index which points to data
14. Compute time of arrival *T* of data bucket
15. Go into doze mode till *T*
16. Wake up at T and access data, *found* = true
17. **end else**
18. end if
19. end While

6. PERFORMANCE EVALUATION

Conservation of energy is the main concern when we try to access data from wireless broadcast. An efficient scheme should allow the mobile device to access its required data by staying active for a minimum amount of time. This would save considerable amount of energy. Since items are distributed based on types and are mapped to suitable locations, we argue that our broadcast deals with clustered data types. The mobile unit has to access a larger major index and a relatively much smaller minor index to get information about the time of arrival of data. This is in contrast to the exponential scheme where the indexes are of equal sizes. The example discussed and Algorithm 1 reveals that to access any data, we need to access the major index only once followed by one or more accesses to the minor index. The number of minor index access depends on the number of internal locations. As the number of internal locations vary for item to item (for example, Weather is generally associated with a City whereas traffic is granulated up to major and minor roads of a city), we argue that the structure of the location mapped information may be visualized as a *forest* which is a collection of *general trees*, the number of general trees depending on the types of information broadcasted and depth of a tree depending on the granularity of the location information associated with the information.

For our experiments, we assume the *forest* as a collection of balanced M-ary trees. We further assume the M-ary trees to be full by assuming the presence of dummy nodes in different levels of a tree.

Thus, if the number of data items is d and the height of the tree is m, then

n= (m*d-1)/(m-1) where n is the number of vertices in the tree and

i= (d-1)/(m-1) where i is the number of internal vertices.

Tuning time for a data item involves 1 unit of time required to access the major index plus time required to access the data items present in the leaves of the tree.

Thus, tuning time with d data items is $t = \log_m d+1$

We can say that tuning time is bounded by $O(\log_m d)$.

We compare our scheme with the distributed indexing and exponential scheme. We assume a flat broadcast and number of pages varying from 5000 to 25000. The various simulation parameters are shown in Table 1.

Figure 5-8 shows the relative tuning times of three indexing algorithms, ie, the LBIS, exponential scheme and the distributed tree scheme. Figure 5 shows the result for number of internal location nodes = 3. We can see that LBIS significantly outperforms both the other schemes. The tuning time in LBIS ranges from approx 6.8 to 8. This large tuning time is due to the fact that after reaching the lowest minor index, the MU may have to access few buckets sequentially to get the required data bucket. We can see that the tuning time tends to become stable as the length of broadcast increases. In figure 6 we consider m= 4. Here we can see that the exponential and the distributed perform almost similarly, though the former seems to perform slightly better as the broadcast length increases. A very interesting pattern is visible in figure 7. For smaller broadcast size, the LBIS seems to have larger tuning time than the other two schemes. But as the length of broadcast increases, it is clearly visible the LBIS outperforms the other two schemes. The Distributed tree indexing shows similar behavior like the LBIS. The tuning time in LBIS remains low because the algorithm allows the MU to skip some intermediate Minor Indexes. This allows the MU to move into lower levels directly after coming into active mode, thus saving valuable energy. This action is not possible in the distributed tree indexing and hence we can observe that its tuning time is more than the LBIS scheme, although it performs better than the exponential scheme. Figure 8, in contrast, shows us that the tuning time in LBIS, though less than the other two schemes, tends to increase sharply as the broadcast length becomes greater than the 15000 pages. This may be attributed both due to increase in time required to scan the intermediate Minor Indexes and the length of the broadcast. But we can observe that the slope of the LBIS curve is significantly less than the other two curves.

Table 1 Simulation Parameters

P	Definition	Values
N	Number of data Items	5000 - 25000
m	Number of internal location nodes	3, 4, 5, 6
B	Capacity of bucket without index (for exponential index)	10,64,128,256
i	Index base for exponential index	2,4,6,8
k	Index size for distributed tree	8 bytes

The simulation results establish some facts about our location based indexing scheme. The scheme performs better than the other two schemes in terms of tuning time in most of the cases. As the length of the broadcast increases, after a certain point, though the tuning time increases as a result of factors which we have described before, the scheme always performs better than the other two schemes. Due to the prescribed limit of the number of pages in the paper, we are unable to show more results. But these omitted results show similar trend as the results depicted in figure 5-8.

7. CONCLUSION AND FUTURE WORK

In this paper we have presented a scheme for mapping of wireless broadcast data with their locations. We have presented an example to show how the hierarchical structure of the location tree maps with the data to create LDD. We have presented a scheme called LBIS to index this LDD. We have used the containment property of LDD in the scheme that limits the search to a narrow range of data in the broadcast, thus saving valuable energy in the device. The mapping of data with locations and the indexing scheme will be used in our DAYS project to create the push based architecture. The LBIS has been compared with two other prominent indexing schemes, i.e., the distributed tree indexing scheme and the exponential indexing scheme. We showed in our simulations that the LBIS scheme has the lowest tuning time for broadcasts having large number of pages, thus saving valuable battery power in the MU.

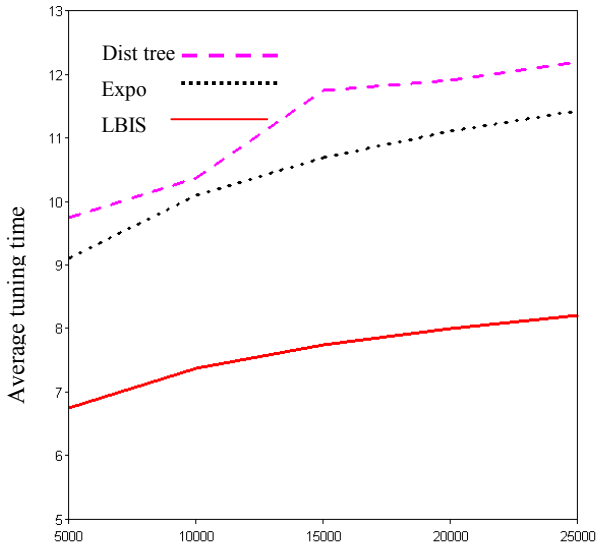

Figure 5. Broadcast Size (# buckets)

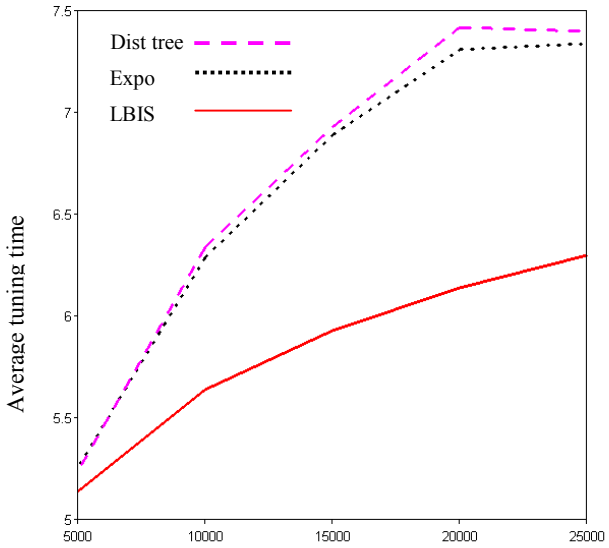

Figure 6. Broadcast Size (# buckets)

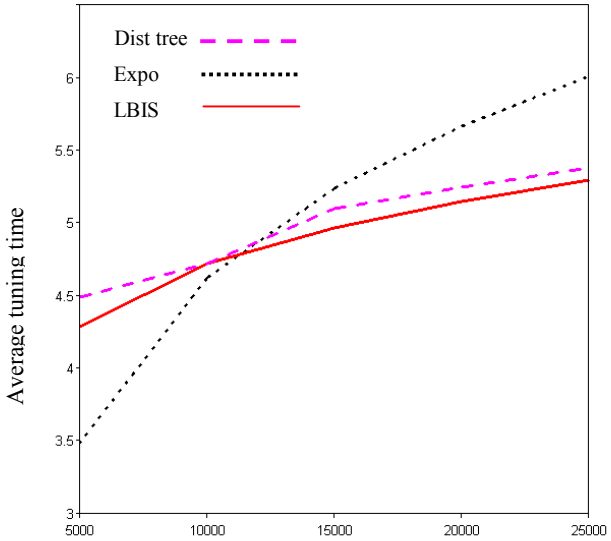

Figure 7. Broadcast Size (# buckets)

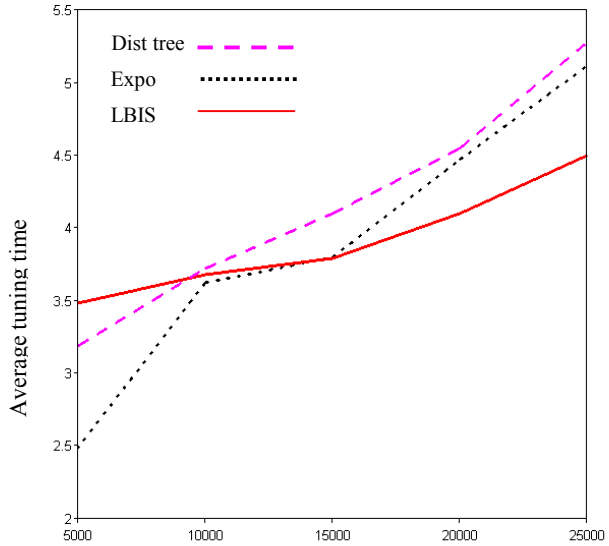

Figure 8. Broadcast Size (# buckets)

In the future work we try to incorporate pull based architecture in our DAYS project. Data from the server is available for access by the global users. This may be done by putting a request to the source server. The query in this case is a global query. It is transferred from the user's source server to the destination server through the use of LEO satellites. We intend to use our LDD scheme and data staging architecture in the pull based architecture. We will show that the LDD scheme together with the data staging architecture significantly improves the latency for global as well as local query.

8. REFERENCES

[1] Acharya, S., Alonso, R. Franklin, M and Zdonik S. Broadcast disk: Data management for asymmetric communications environments. In Proceedings of ACM SIGMOD Conference on Management of Data, pages 199–210, San Jose, CA, May 1995.

[2] Chen, M.S.,Wu, K.L. and Yu, P. S. Optimizing index allocation for sequential data broadcasting in wireless mobile computing. IEEE Transactions on Knowledge and Data Engineering (TKDE), 15(1):161–173, January/February 2003.

[3] Hu, Q. L., Lee, D. L. and Lee, W.C. Performance evaluation of a wireless hierarchical data dissemination system. In Proceedings of the 5th Annual ACM International Conference on Mobile Computing and Networking (MobiCom'99), pages 163–173, Seattle, WA, August 1999.

[4] Hu, Q. L. Lee, W.C. and Lee, D. L. Power conservative multi-attribute queries on data broadcast. In Proceedings of the 16th International Conference on Data Engineering (ICDE'00), pages 157–166, San Diego, CA, February 2000.

[5] Imielinski, T., Viswanathan, S. and Badrinath. B. R. Power efficient filtering of data on air. In Proceedings of the 4th International Conference on Extending Database Technology (EDBT'94), pages 245–258, Cambridge, UK, March 1994.

[6] Imielinski, T., Viswanathan, S. and Badrinath. B. R. Data on air – Organization and access. IEEE Transactions on Knowledge and Data Engineering (TKDE), 9(3):353–372, May/June 1997.

[7] Shih, E., Bahl, P. and Sinclair, M. J. Wake on wireless: An event driven energy saving strategy for battery operated devices. In Proceedings of the 8th Annual ACM International Conference on Mobile Computing and Networking (MobiCom'02), pages 160–171, Atlanta, GA, September 2002.

[8] Shivakumar N. and Venkatasubramanian, S. Energy-efficient indexing for information dissemination in wireless systems. ACM/Baltzer Journal of Mobile Networks and Applications (MONET), 1(4):433–446, December 1996.

[9] Tan K. L. and Yu, J. X. Energy efficient filtering of non uniform broadcast. In Proceedings of the 16th International Conference on Distributed Computing Systems (ICDCS'96), pages 520–527, Hong Kong, May 1996.

[10] Viredaz, M. A., Brakmo, L. S. and Hamburgen, W. R. Energy management on handheld devices. ACM Queue, 1(7):44–52, October 2003.

[11] Garg, N. Kumar, V., & Dunham, M.H. "Information Mapping and Indexing in DAYS", 6th International Workshop on Mobility in Databases and Distributed Systems, in conjunction with the 14th International Conference on Database and Expert Systems Applications September 1-5, Prague, Czech Republic, 2003.

[12] Acharya D., Kumar, V., & Dunham, M.H. InfoSpace: Hybrid and Adaptive Public Data Dissemination System for Ubiquitous Computing". Accepted for publication in the special issue of Pervasive Computing. Wiley Journal for Wireless Communications and Mobile Computing, 2004.

[13] Acharya D., Kumar, V., & Prabhu, N. Discovering and using Web Services in M-Commerce, Proceedings for 5th VLDB Workshop on Technologies for E-Services, Toronto, Canada,2004.

[14] Acharya D., Kumar, V. Indexing Location Dependent Data in broadcast environment. Accepted for publication, JDIM special issue on Distributed Data Management, 2005.

[15] Flinn, J., Sinnamohideen, S., & Satyanarayan, M. Data Staging on Untrusted Surrogates, Intel Research, Pittsburg, Unpublished Report, 2003.

[16] Seydim, A.Y., Dunham, M.H. & Kumar, V. Location dependent query processing, Proceedings of the 2nd ACM international workshop on Data engineering for wireless and mobile access, p.47-53, Santa Barbara, California, USA, 2001.

[17] Xu, J., Lee, W.C., Tang., X. Exponential Index: A Parameterized Distributed Indexing Scheme for Data on Air. In Proceedings of the 2nd ACM/USENIX International Conference on Mobile Systems, Applications, and Services (MobiSys'04), Boston, MA, June 2004.