LETTER

An Index Based on Irregular Identifier Space Partition for Quick Multiple Data Access in Wireless Data Broadcasting*

SeokJin IM**(a), Nonmember and HeeJoung HWANG††(b), Member

SUMMARY This letter proposes an Index based on Irregular Partition of data identifiers (IIP), to enable clients to quickly access multiple data items on a wireless broadcast channel. IIP improves the access time by reducing the index waiting time when clients access multiple data items, through the use of irregular partitioning of the identifier space of data items. Our performance evaluation shows that with respect to access time, the proposed IIP outperforms the existing index schemes supporting multiple data access.

key words: wireless data broadcast system, air index, irregular space partition, multiple data access

1. Introduction

A wireless data broadcast system is a promising way of delivering information to an arbitrary number of clients in bandwidth-limited communication environments. In the system, a broadcast server periodically disseminates data items, such as stock market data, over a wireless channel in buckets, the smallest logical broadcasting unit. After tuning into the channel, an arbitrary number of clients can independently search and download their desired items. For example, a client can find stock prices of Microsoft, Google and Apple from the wireless broadcast channel devoted to the stock market. After tuning into the channel, constantly listening to the channel until the desired content appears, the client can then download it [1]–[3]. Thus, wireless data broadcast has the merit of scalability: the server can support a great number of clients simultaneously accessing data items, without needing to expand the physical capacity of the server though the rapid increase of the clients. Currently, practical services in wireless data broadcast are available through the DAB/DAB+ system [10] and ADS-B system [11]. In particular, the DAB+ system Journaline service efficiently disseminates text-based information, such as traffic and weather reports, sports events and results, advertisements, and stock market information.

Wireless data broadcast introduces an air index to allow clients to selectively access the items they wish to download on the channel, without needing to listen continuously. An air index maintains unique identifiers of data items and time pointers to them, their broadcasting times. The server disseminates the index by interleaving it with data items on the channel. The clients use the index to find their desired items and the time pointers to them. Thus, using the time pointers, the clients can selectively access only the desired data items on the channel [4], [5].

The access time is the time duration from the time a client tunes in to the broadcast channel, to the time the client finishes downloading all of their desired items. The access time consists of initial probe, index waiting time, and the time for searching and downloading data items. In the initial probe, the clients obtain the time pointer to the first index from the bucket they first meet after tuning in. Here, each bucket carries the time pointer to the first index appearing after itself. Then the clients wait for the index in sleep mode, and wake up in active mode at the time pointer, and access the index. Through accessing the indexes, the clients search time pointers to their desired items. Using the pointers, the clients selectively download the items from the channel. To improve the access time, it is important to reduce the size of the index because the size lengthens the length of a broadcast cycle, called Bcast. The index waiting time should also be reduced. The index waiting time is affected by the spaces between interleaved indexes on the channel. In general, keeping the spaces regular makes the best index waiting time when the interleaving number of the indexes is fixed.

Related Works. For clients’ efficient data access, various indexes have been proposed, using integers as the unique identifier of data items. Here, we review the existing indexes to enable clients to access multiple data in a search, through an access to the indexes. Multiple data access in a search is common and realistic, because there is no restriction on the number of data items for the clients to be able to access in a search [7]–[9]. For example, using a wireless data service such as the Journaline service on DAB+ systems, the clients can access multiple data items in a search. Through the wireless data broadcast service disseminating sports events information with an index to support multiple data access, the clients can access 3 sports events information in a search, e.g., soccer, baseball, and basketball.

Exponential index (EXP) is a distributed indexing table that is allocated to each data item [4]. EXP for a data items holds identifiers and time pointers of items that are separate from the item in exponential distance on the channel. A
data item and its indexing table are combined in units of chunks, then the chunks are broadcast. EXP enables the clients to access multiple data on the channel with an index table holding multiple time pointers in itself. However, the access time by EXP rapidly increases, due to the increase of the size of EXP according to the increment of data items.

GDI is a distributed indexing table that is allocated to groups of data items, unlike EXP [5]. GDI regularly partitions the identifier space (IS) of data items to disjoint whole data items into groups. GDI is broadcast prior to the data group allocated to it, and can support the clients accessing multiple data on the channel by time pointers in an indexing table. However, when the identifier distribution is skewed over IS, regular partition of IS makes different the number of data items that each group has, rather than uniform. This makes the spaces between indexing tables on the channel, and deteriorates the access time.

**Our Contribution.** We propose an Index based on Irregular Identifier space Partition (IIP) to support the clients quickly accessing multiple data on the channel in a wireless data broadcast environment. IIP is an indexing table for a group of data items, and is broadcast prior to the group in the channel. IIP targets improvement of the access time, although the data distribution over IS is skewed.

IIP achieves this goal by reducing the index waiting time and its size. For the improved index waiting time, we equalize the spaces between successive indexing tables, using disjointed groups that have the same number of items. For the same sized groups, we partition IS irregularly. Thus, IIP keeps equal the spaces between indexing tables, notwithstanding the skewed data distribution over IS. This irregular partition makes IIP different from GDI, which uses regular partition and deteriorates the indexing waiting time in skewed data distribution over IS. IIP contributes to improving the access time by reducing the index waiting time for an index after the initial probe, through the equal spaces between indexes.

IIP introduces parent groups and child groups by a two-leveled irregular space partition, to which its own index is allocated, named a parent and child index. The information on irregular partition and data items is carried efficiently using the two kinds of indexes. To support clients’ linear channel access pattern, the indexes are organized in table form. Also, the indexes distributed in equal spaces provides multiple access paths to data items on the channel.

The rest of the letter is organized as follows. Section 2 presents the proposed IIP. Section 3 describes simulation environments, and then evaluates their performance. Finally, Sect. 4 concludes the letter.

**2. Irregular ID Partition-Based Indexing Scheme**

We consider the environment where the clients search the answers to their own multipoint queries, $Q_k$ defined below, on the wireless channel, the server broadcasting the $N$ data items of dataset $D$.

The item $d_i$ of $D$ has non-negative integer $i$ as its unique identifier. For the dataset $D$, identifier space IS is defined as the set of integers from 0 to the biggest identifier of the items of $D$. For example, Fig. 1 shows that for dataset $D = \{d_1, d_2, d_5, d_8, d_9, d_{10}, d_{11}, d_{15}, d_{17}, d_{19}, d_{21}, d_{27}, d_{28}, d_{31}, d_{53}, d_{63}\}$, IS is $0 \leq IS \leq 63$.

**Definition 1:** Multipoint Query $Q_k$; $Q_k$ is a set of integer identifiers of $k$ data items to search on the channel.

When a client processes $Q_3 = \{2, 10, 63\}$, the client searches and downloads data items $d_2$, $d_{10}$, and $d_{63}$ from the channel.

In order to organize the proposed IIP, we disjoin $N$ data items into two-leveled groups, the parent data group and the child data group. After sorting $D$ in ascending order of identifiers, we set every $n_p = \lceil N/r_p \rceil$ data items as a parent data group, where $r_p$ is the size factor for the parent data group. Then, every $n_c = \lceil n_p/r_c \rceil$ data items in a parent data group is split up into a child data group, where $r_c$ is the size factor for the child data group. Figure 1(a) shows the parent and child data groups at $r_p = 0.25$ and $r_c = 0.5$. Parent group $P_i$ and its child group $C_j$ for corresponding data groups are defined in the way of irregular IS partitioning as below:

**Definition 2:** Parent group $P_i$. $P_i$ is the tuple of the minimum and the maximum identifiers that contain the $i$-th parent data group, and is represented as below.

$$P_i = \langle id^M_i, id^m_i \rangle, \text{ for } 0 \leq i < \lceil 1/r_p \rceil$$

Here, $id^M_i$ is the maximum identifier of the $i$-th parent group, and $id^m_i$ is the minimum identifier of the parent group with the value of $id^M_i + 1$.

**Definition 3:** Child group $C_j$. $C_j$ is the tuple of the minimum and the maximum identifiers that cover the $j$-th child data group in $P_i$, and is represented as below.

$$C_j = \langle id^M_j, id^m_j \rangle, \text{ for } \lceil 1/r_p \rceil \cdot i \leq j < \lceil 1/r_p \rceil \cdot (i + 1)$$

Here, $id^M_j$ is the maximum identifier of the $j$-th child data group, and $id^m_j$ is the minimum identifier of the child group with the value of $id^M_{j-1} + 1$.

Figure 1 (a) shows parent and child data groups of $D$, as well as parent and child groups corresponding to data groups. Figure 1(b) depicts the skewed identifier distribution and the two-leveled hierarchical groups by irregular partition.
2.1 Index Structure

We organize the proposed IIP with two kinds of linear indexing tables: parent index PI for P_i, and child index CI_j for C_j.

PI for P_i is configured as follows:

\[ PI = \{P_i, CGT, PGT\} \]  \hspace{1cm} (3)

- \( P_i \): the tuple representing parent group \( P_i \).
- \( CGT \) (Child Group Table): \( \{C_j, t_j\} \), where \( C_j \) is a child group in \( P_i \), and \( t_j \) is the time pointer to \( CI_j \) for \( C_j \) for \( [1/r_p] \cdot i < j \leq [1/r_p] \cdot (i + 1) \).
- \( PGT \) (Parent Group Table): \( \{P_a, t_a\} \), where \( P_a \) is a parent group, and \( t_a \) is the time pointer to \( PI_a \) for \( P_a \) for \( 0 \leq a < [1/r_p] \).

\( P_i \) informs the clients of the identifier coverage of \( PI_i \). Using \( P_i \), the clients can decide whether they access the child groups of \( P_i \) or not. The table \( CGT \) enables the clients to obtain the local view of the identifier partition, i.e., the coverage of each child group of \( P_i \), and the time pointer to the child group. Using \( CGT \), the clients access child groups containing identifiers in a given multi-point query \( Q_k \). The table \( PGT \) provides the clients with a global view of the identifier partition, and the time pointers to parent groups on the channel.

\[ CI_j = \{IDT, NCG, t_{nc}, t_{np}\} \]  \hspace{1cm} (4)

- \( IDT \) (ID Table): \( \{id, t_d\} \), where \( id \) is the identifier of a data item in \( C_j \), and \( t_d \) is the time pointer to the data on the wireless channel.
- \( NCG \) (Next Child Group): the tuple of the next child group of \( C_j \).
- \( t_{nc} \): the time pointer to the next child group of \( C_j \).
- \( t_{np} \): the time pointer to the next parent group of \( C_j \).

The table \( IDT \) enables the clients to extract identifiers within a given multipoint query \( Q_k \), and the time pointers to the identifiers. \( NCG \) informs the clients of the coverage of the next child group \( C_{j+1} \), and \( t_{nc} \) enables the clients to access \( C_{j+1} \). \( NCG \) and \( t_{nc} \) construct the chain connecting all the child groups on the channel. Using \( NCG \) and \( t_{nc} \), the clients access all the child groups on the channel, one after another. The time pointer \( t_{np} \) supports the clients accessing the next parent index of \( CI_j \), and obtaining the global information on the partition of IS.

2.2 Channel Structure

Using the proposed IIP, the wireless channel is organized as follows. Figure 2 shows that every parent index is followed by all the child indexes, and the data items in the child groups in the increasing order of group number.

Figure 2 shows the channel structure by the proposed IIP for the dataset \( D \) in Fig 1. The proposed scheme enables the spaces between successive indexes on the channel to be equalized. That makes reduced the index waiting time on the channel, and improves the access time.

2.3 Multipoint Query Processing

Using the parent and child indexes of the proposed IIP, a client can process its own multipoint query \( Q_k \). The client begins the process with access to the first encountered index on the channel. For the processing, the client uses two priority queues, \( gpQueue \) and \( dpQueue \). These are queues to hold the time pointers to indexes and data items in increasing order, respectively.

Algorithm 1 depicts the procedure for processing a multipoint query using the proposed IIP. When accessing parent index \( PI_i \) for \( P_i \) in lines 6 and 7, \( gpQueue \) is updated by function \( getTimeFromCGT() \) and \( getTimeFromPGT() \), as Functions 1 shows. With child index \( CI_j \) for \( C_j \) in lines 9 and 10, \( dpQueue \) is updated by function \( getTimeFromCI() \) and \( extractDataWithCI() \) in Functions 1, respectively.

3. Performance Evaluation

3.1 Simulation Environment

For the evaluation, we implemented a simulation testbed
that consists of a broadcast server, a wireless channel, and a client, using the discrete time simulation package, SimJava. Because in a wireless data broadcast, clients do not affect each other, we implemented one client in the testbed. In simulation using the testbed, we measured the access time by the proposed IIP, GDI, and EXP in the unit of buckets.

For the simulation, we used a real dataset of 5922 cities of Greece [6]. After applying the Hilbert Curve to the 2D space containing the locations of cities, we used the Hilbert Curve value for the location of each city as the unique identifier of the city. Thus we can obtain natural skewed identifier distribution.

As the parameters for simulations, we set both the identifier size and the time pointer size to 8 bytes. We set the size of a data item to 1024 bytes, and the bucket size to 64 bytes. Also we set \( r_p \) and \( r_c \) for the parent and child group to 0.01 and 0.1, respectively. We apply the same value of \( r_p \) and \( r_c \) to GDI for fair comparison. Also for fairness, we set the base of EXP to 2, to provide the client with the most time pointers in an indexing table of EXP. We set \( k \) of \( Q_k \) to 1, 10, 20, 40, and 60. For each \( k \), we ran 50,000 multipoint queries, and measured the average access time.

### 3.2 Comparison of the Performance

In order to evaluate the performance of the proposed IIP, we compare \( Bcast \), the length of a broadcast cycle, and the access time by IIP, GDI, and EXP that support the client to process multipoint query, \( Q_k \), on the broadcast channel.

Figure 3 (a) shows that \( Bcast \) by IIP and GDI based on grouping are shorter than that by EXP. This reveals group-based indexing is more efficient for the client to access queried data items. The shorter \( Bcast \) results from the smaller size of IIP and GDI, than that of EXP.

![Figure 3](image_url)

**Figure 3** Comparison of IIP with other schemes in the index size and access time.

Figure 3 (b) shows the access time of the proposed IIP, GDI, and EXP in the number of buckets, when the client processes single point queries, \( Q_k \), \( k = 1 \), and multipoint queries, \( Q_k \), \( k \) = 10, 20, 40 and 60. This depicts that in the case of the access to multiple data, as well as to a single datum, the proposed IIP is superior to GDI and EXP. Also, it demonstrates that IIP helps the client to quickly access the queried data items through the reduced index waiting time by the equal spaces among successive indexes on the channel, from irregular ID space partition over skewed identifier distribution. Our evaluation shows that the reduced index waiting time by the proposed IIP is effective in reducing the access time not only for multiple data, but also for a single datum.

### 4. Conclusion

We propose an index based on irregular identifier space partition to enable the clients to process time-efficient multipoint queries on a wireless data broadcast channel, where the distribution of identifiers of data items is skewed over the identifier space. Aimed at supporting the clients quickly accessing multiple data items, the proposed IIP partitions the data through irregular space partition into two-leveled groups. This allows the client’s index waiting time on the channel to be reduced, by virtue of the equal distances among successive indexes. By simulation, we demonstrate the proposed IIP outperforms the existing indexes for multiple data access.

### References


