The Replication in Varying Fanout Indexing Technique for Skewed Access Patterns in the Wireless Mobile Environments

Mani Pandey
Research Scholar
AKGEC, Ghaziabad

Vikas Goel
Assistant Professor
AKGEC, Ghaziabad

ABSTRACT

Due to limited battery power, the most important issue in mobile computing is energy saving. The energy efficiency can be achieved through indexed data organization over wireless channels. In this paper, we explore the balanced and imbalanced index tree with varying fanout over skewed data. We purpose a varying fanout indexing technique with replication for skewed data over a single wireless communication channel. The proposed indexing technique is compared and analyzed with the fixed fanout indexing technique for skewed data over a single wireless channel. The result shows the decrease in directory miss and the depth of the tree is also reduced.

Keywords: Varying Fanout, skewed data, balanced index tree, Imbalance Index tree, and directory miss.

1. INTRODUCTION

In recent years, the utilization of wireless technology devices has been growing at an exponential rate. For wireless data applications, the data dissemination methods are categorised between the two: point-to-point access and broadcast. In point-to-point access, a logical channel is established between the client and the server, where, queries are submitted to the uplink of server and returns the results to the client as in a wired network. In broadcast, data is transmitted simultaneously to all users who are residing in the broadcast area. It is to choice of a client to select the data it wants [1]. Data broadcasting is referred to as broadcasting which mainly transmits data, for example characters, shapes, still pictures, images, sounds etc., and differs from television broadcasting which mainly transmits videos or radio broadcasting which transmits sounds only [13]. But there is problem lies with data broadcast, when a mobile client retrieve a data item, it has to continuously monitor the broadcast channel until the data item of its interest arrives. This will consumes a lot of battery power. The limited battery capacity of mobile client's device makes power conservation a critical issue in the design of broadcast systems. It is important for mobile clients for energy saving will operate in two different modes**:** active mode and doze mode. The mobile clients can retrieve data from broadcast channels in the active mode only. However, the clients have much higher rates of battery consumption in the active mode than in the doze mode. The wireless devices can stay in the power saving mode or doze mode and tune into the broadcast channel only when the data items of interest to them arrive, hence lots of energy of these devices can be saved [1].The efficiency of the broadcast channels is estimated by two criteria used frequently :access latency and tuning time. The access latency refers to how fast the client can access the requested data and tuning time refers to the duration for which the client stays active to receive the requested data items[13] .

The performance of broadcast systems is always characterized by these two metrics. The tuning time can be reduced by means of air indexing. So by adding index information to the broadcast file one can save mobile device power battery. Without indexing, the clients have to be continuously active and monitor the broadcast channel until the requested data item would arrive. This consumes significant amount of battery power and sacrifices energy efficiency. So the issue is to save battery power with minimized access latency during data broadcast in the single channel where data and index can broadcasted in the same channel.

The broadcast disks in [7] takes into consideration for non-uniform data access distribution. In this approach, the several data items with similar access rates are grouped together to form logical disks. Each disk is assigned a relative broadcast frequency, more popular items are assigned higher frequencies. The broadcast schedule is then constructed by circularly picking up items from the disks based on their relative broadcast frequencies. Another indexing technique proposed in [9], a signature-based indexing method. Specifically, a broadcast cycle is divided into a number of frames. Each frame is preceded by a signature of its data item in the broadcast schedule. This allows the client to check whether a requested item is in the frame by investigating the signature only. However, this signature does not provide the arrival times of data items. Thus, when a match is found in a signature, the data items which are indexed by the signature have to be searched sequentially. Moreover, since a signature does not contain global information about the broadcast, data accesses require sequential scans of signatures. In [1] authors applied the tree-based index designed for traditional disk storage to wireless data broadcast. The index nodes in the tree are interleaved with data items in the broadcast schedule. Starting from the root index node, the client follows the links in the tree and tunes to selected index nodes to locate the requested item. The tree-based indexes are extended in [10, 11] by constructing multiple index trees that share links. The resultant index structure allows searching to start at anywhere in the broadcast. Unfortunately, most tree-based indexes are applicable to flat broadcast only because they require data items be ordered by their key values in the broadcast schedule. Besides from tree-based indexes, hash functions can also be used for indexing purpose to map data items to the slots in the broadcast schedule [8]. A salient feature of hash-based index is that it eliminates the need to broadcast index structures, since only a hash function is broadcast together with data. While the broadcast overhead of a tree based index structure normally increases with the number of data items, the broadcast overhead of a hash function is largely independent of the latter. Another energy efficient indexing scheme called MHash that optimizes tuning time and access latency in an integrated fashion [14].

It is noted that, in most databases, the access frequencies of different data items are usually different from one another [2]. Among the selective tuning strategies, in [3] constant fanout (CF) and variant fanout (VF) index tree takes the access probabilities of data items into consideration. More popular data item may be frequently accessed by the mobile clients than the less popular ones. This is known as skewed data access. However, VF assumes the sorted data items according to the access probabilities and index tree is constructed according to this sorted order. But in real life applications, the index tree should be constructed according to the key values of the data items, not according to access probabilities. The Alphabetic Huffman tree [4],[5] preserve leaf ordering on any input sequence used to construct them(similar to B+ Trees) ,i.e., left-to-right scan of the leaves of each tree will show the leaves ordering by their keys, and function as search trees. In order to minimize the tuning time, we consider two cases : one for fixed index fanout, and one for variant index fanout considering k-ary Alphabetic Huffman tree construction. For the case of fixed index fanout ,in light of Alphabetic Huffman tree construction, we consider binary fanout. And for the case of variant index fanout we construct the k-ary Alphabetic Huffman tree which will produce varying fanout (between 2 and k). In VF , the replication of index nodes would not be considered. That means mobile clients always have to wait for the next cycle to traverse the index tree to get the requested data, resulting in the increase of the access time. In this paper we will consider the replication of index nodes at fixed level of tree in both fixed index fanout tree and variant fanout tree.

2. BACKGROUND

In the wireless communication environment, a broadcast cycle consists of a collection of data items which are broadcasted cyclically on the wireless channel. The mobile client in the broadcast area listens to the channel to retrieve the data item of their interest. This is known as selective tuning [1]. If the data is broadcasted without any index, the mobile client will have to listen to the wireless channel, on the average, half of the total broadcasting time for the complete file. Hence by using proper indexing this selective tuning allows mobile clients to stay active only when the data of interest is present , thereby saving lot of battery consumption. In this section we will talk about the balanced and imbalanced index tree techniques.

2.1 Balanced Index Trees

Most of the prior work is on symmetric balanced index tree with all leaves are in the same level and essentially the same fanouts for all index nodes. B+ tree indexing is a widely used indexing technique in traditional disk-based environments. It is also one of the first indexing techniques applied to wireless environments. The use of B+ tree indexing in wireless environments is very similar to that of traditional disk based environments [15]. Indices are organized in B+ tree structure to accelerate the search processes. In [1] two indexing schemes based on B+ tree data structure, (1, m) indexing and distributed indexing, are presented and assumed as balanced index tree. In distributed indexing, every broadcast data item is indexed on its primary key attribute. Indices are organized in a B+ tree structure.

2.2 Imbalanced Index Trees

In reality index trees may be imbalanced as the distance of leaf nodes from the root is not same or leaf nodes are not at the same level of the tree. Most examples of these trees are Huffman tree. It has been shown by experimental results that the use of imbalanced index trees will give considerable improvement in performance over the use of balanced trees, and such an advantage becomes even more prominent as the skewness of the data access increases [3].

2.2.1 B^+ - Tree Based Indexing

The B+ Tree based distributed indexing is a technique in which index is partially replicated introduced in [1] provides a method to multiplex it together with the corresponding data file on the broadcast channel. In this section we will consider two different states of B+Tree.

(A)Fixed Fanout

The distributed indexing method is not a new method of index construction but a method of allocation of a file and its index on the broadcast channel. An index tree for four data items is shown in Fig1(a). The algorithm divides the index tree into two parts: the replicated part and the non-replicated part. The algorithm replicates only the replicated part (R), and the number of times each node appears in that part equals the number of its child nodes [1]. Moreover, each index node in the replicated part has the control index used to direct clients to a proper branch (a higher-level index node) in the index tree.

On the other hand, each node in the non-replicated part will appear only once in front of the set of data nodes it indexes. The distributed indexing traverses the index tree and allocates the index nodes and the data nodes to buckets in a broadcast cycle. The index node R is broadcasted first. Next, the sub tree rooted by index node A1 is traversed in preorder, resulting in < A1, 1, 2>. After that, since the root node R is in the replicated part, this node is broadcast again. Furthermore, traversal sequence < A2, 3, 4> of the sub tree rooted by index node A2 is appended to the broadcast cycle. Each data bucket contains the offset to the nearest-replicated index bucket [12].

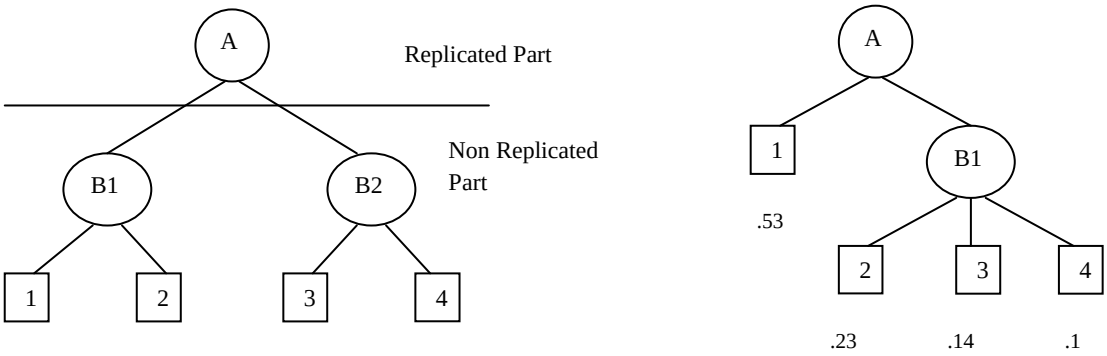

Fig. 1(a) A distributed index tree (b) A VF index tree

(B) Varying Fanout

In [3], Chen proposed the variant fanout for the wireless broadcast. Fig 1(b) illustrates the Varying fanout index tree of four data items. The VF assumes that the broadcast data items are sorted according to the descending order of their access probabilities. The algorithm first attaches all the data items to the root node A, then after some evaluation, VF groups nodes with small access probabilities and moves them down to next level [3]. Finally the index node B1 is attached to the root A according to descending order of access probability of data item and the same grouping process is proceeded to the root node A. Fig 1(b) shows the final VF index tree which is imbalanced.

2.2.2 Huffman Based Indexing

The Huffman tree construction considers the minimum frequency sum so that the index pointers of more popular data items are higher up in the tree than others. However , the Huffman tree constructed [4] not a search tree since users will need to know the encoding of a file before they can traverse the tree for the given file. To illustrate we show in Fig 2. a regular 3–ary Huffman tree constructed from Table 1[4]. We can see from Fig. 2 that the index pointers are at different levels of the tree based on the popularity information, there is no way of traversing the tree to find a desired pointer by knowing only its key. Hence the only way to access a specific leaf is to know its Huffman code.

However this is a problem for mobile client since they only have the key of the file they are searching for. The mobile client cannot know the Huffman code of the desired file in advance since the code depends on the popularity patterns of other files being broadcasted at that time and may change over time. There exists a special class of Huffman tree known as Alphabetic Huffman tree [4] [5] which function as search tree and preserve the leaf ordering.

Table 1:Files and their popularity patterns

Key	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Frequency	23	4	12	10	17	31	15	21	29	19	7	12	16	14	20

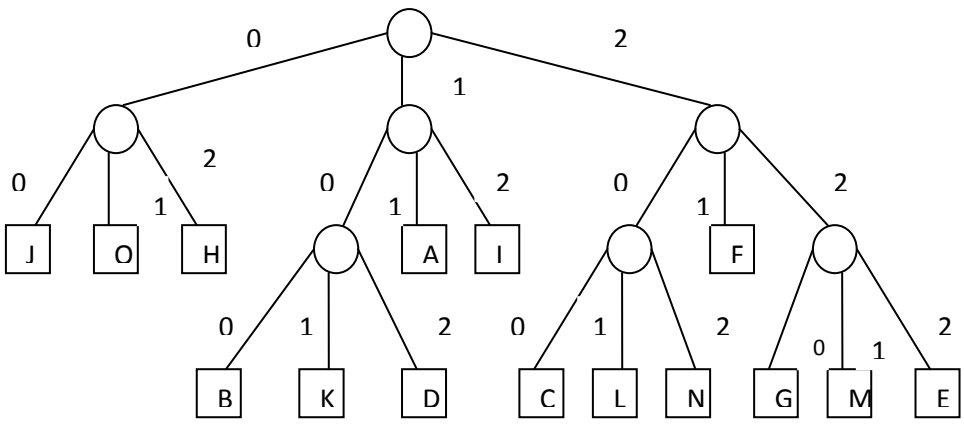

Fig. 2 The Huffman Tree

(A) Fixed Fanout

Firstly, we construct the fixed fanout Alphabetic Huffman index tree considering binary fanout [4, 5] . For k=2 the fanout will be constant. Table2 is an example data set with 20 data items considered in [6], which will be continuously used for fixed and varying fanouts in the paper. The Alphabetic Huffman Tree is constructed in two steps given by Hu & Tucker in [5] shown in Fig 3 and Fig 4(a):

Step1: Start with the initial sequence of data items d_1, d_2, d_3,.....,d_n, having leaf nodes in their given order. Combine the data nodes d_i , d_j such that sum of their frequencies is the minimum and there are no leaves between them and also d_i and d_j are the leftmost nodes among all candidates. A new sequence of data items $d_1, d_2,...,d_{i-1}, d_{(i, j)}$,d_{i+2} ,...., d_n where $d_{(i, j)}$ is an index node and other are still leaf nodes ; now combine some adjacent with minimum frequency in this new sequence and replace the combined pair with sum and so on. This will produce a tree T without alphabetic ordering of the data nodes as in Fig.3 where we record the frequencies of each index node inside the circle as index key values.

Step 2: Now record the level of each data node of T denoted as L_i . Consider the root node level is 1. From bottom to the root, rearrange the pointers such that for each level the leftmost two nodes have the same parent, and then the next two and so on [6]. Therefore Alphabetic Huffman Tree T' is generated without changing the level of each node in T as shown in Fig. 4.

We also extended this algorithm in the next section to construct *k*-ary (Varying Fanout) Huffman-Tree, by merging at most *k* nodes in step1, and combining up to *k* nodes with the same parent in step2. After generating the alphabetic Huffman tree *T'* in Fig 4, we cut *T'* at level *l*, and perform a depth first traversal. The index nodes above *l* are still called control index, and index nodes below *l* is search index.

Table 2. Files and their popularity patterns

Key	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Frequency	23	4	12	10	17	31	15	21	29	19	7	12	16	14	20	48	11	22	18	8

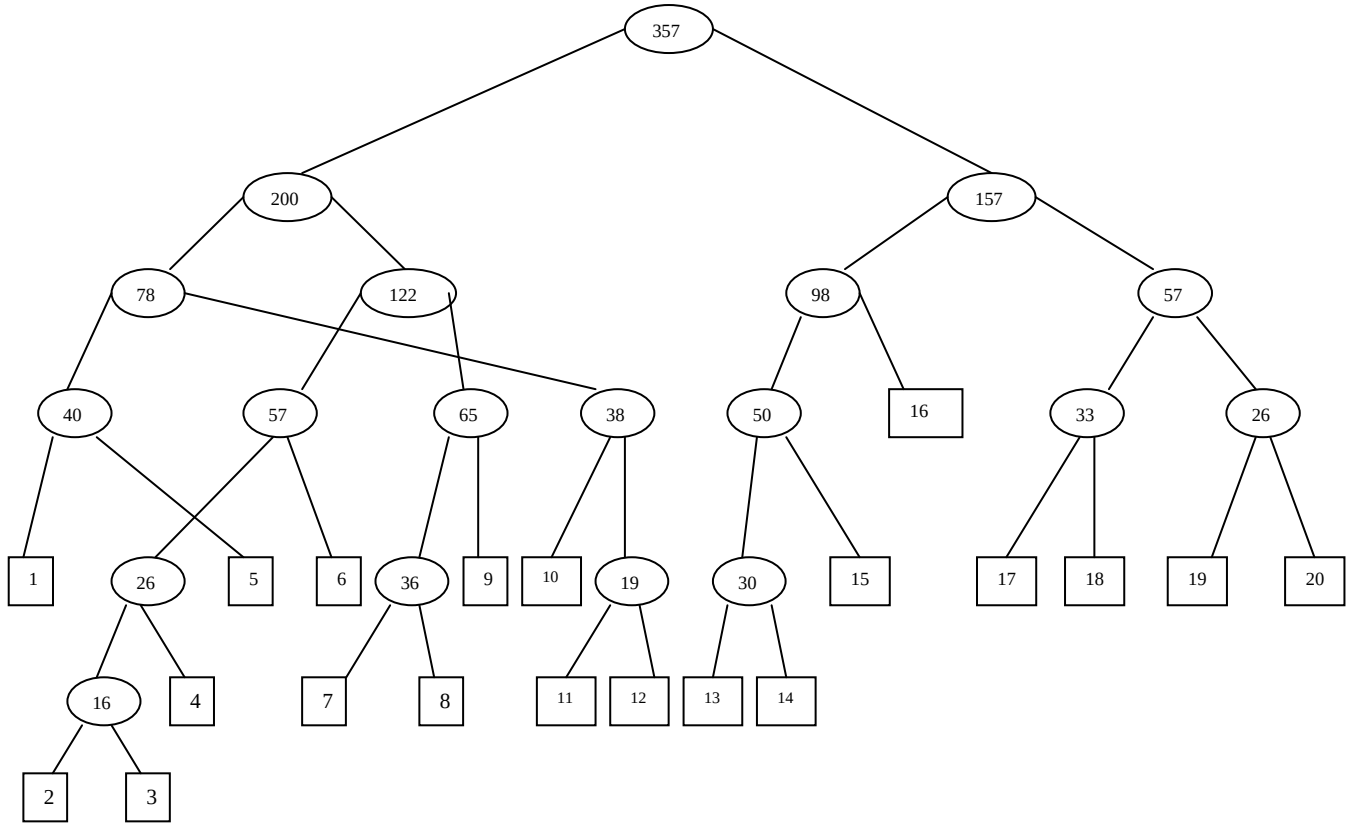

Fig 3 The first step of constructing T

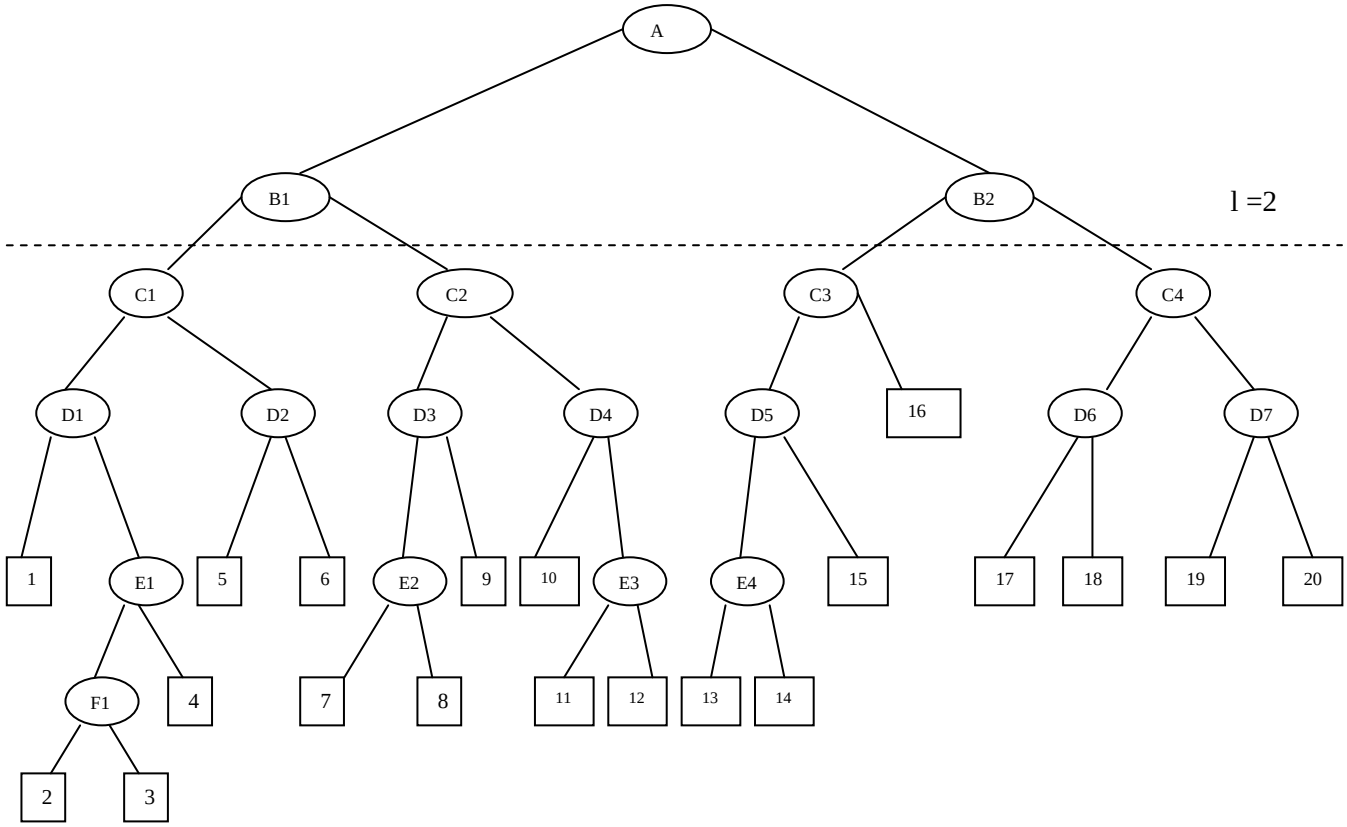

Fig 4 The Final Huffman Tree T'

(B) Varying Fanout

In the modification of above Hu & Tucker algorithm , we allow at most k nodes to be combined into a single super-node during the passes in step1 , instead of two nodes in [5]. We also allow combining k leaf nodes if they are consecutive in the construction sequence, while the other cconditions remain the same. Also, the second step remains the same except that we allow up to k nodes to be join together to have the same parent (i.e. from 2 to k nodes. Since the conditions on combining nodes in the first step are modified minimally, we can still perform the reordering phase similar to that in [5]. In our example, the varying fanout number, k=3. So we can join 2 or 3 nodes to have the same parent in the index tree .The k-ary construction for the Table 2 is given in Fig 5.

3. REPLICATION IN VARYING FANOUT (RVF) INDEX TREE

The proposed replication in varying fanout index tree indexing technique is explained here. The replication at a certain fixed level in index trees would have been considered for varying fanout. The number of times a index node is replicated depends on the number of fanouts it has. we consider the same algorithm for the construction of k-ary Huffman Tree with three data items d_i , d_j ,d_k , such that they are consecutive and their frequency sum is minimum [5, 6]. The constructed alphabetic Huffman tree is a imbalanced index tree. The k-ary Alphabetic Huffman tree is shown in Fig 5, which is having replicated varying fanout.

We cut the tree T'' at level *l* so that the index nodes above *l* is still called control index, and index nodes below *l* is called search index. Now we will perform the depth first traversal of the replicated tree given in Fig 6(a) and hence the final broadcast sequence B generated in Fig 6(b). The index node A is traversed first. Next, the sub tree rooted by B1 is traversed , then B2 is traversed in preorder. Since root node A and its child nodes B1 and B2 are in the replicated part, these nodes are broadcasted again. As root node A is having its two child nodes B1 and B2 will be replicated two times. But the index node B1 is having child nodes 1, C1 and 9, then it will replicated three times and B2 will also replicate three because it has its child nodes as C2, C3, C4.The important feature of this k-ary index trees is that we may end up with a tree with smaller depth resulting in smaller broadcast sequence. It is important to note that in the k-ary construction, it may not be always possible (or optimal) to combine k nodes together. Therefore this k-ary Alphabetic Huffman tree will have a fanout that varies between 2 and k.This indexing technique would solve the problem of directory miss which would be occurring in previous VF because replication is not considered. Moreover this technique would provide good result in terms average access time and tuning time also

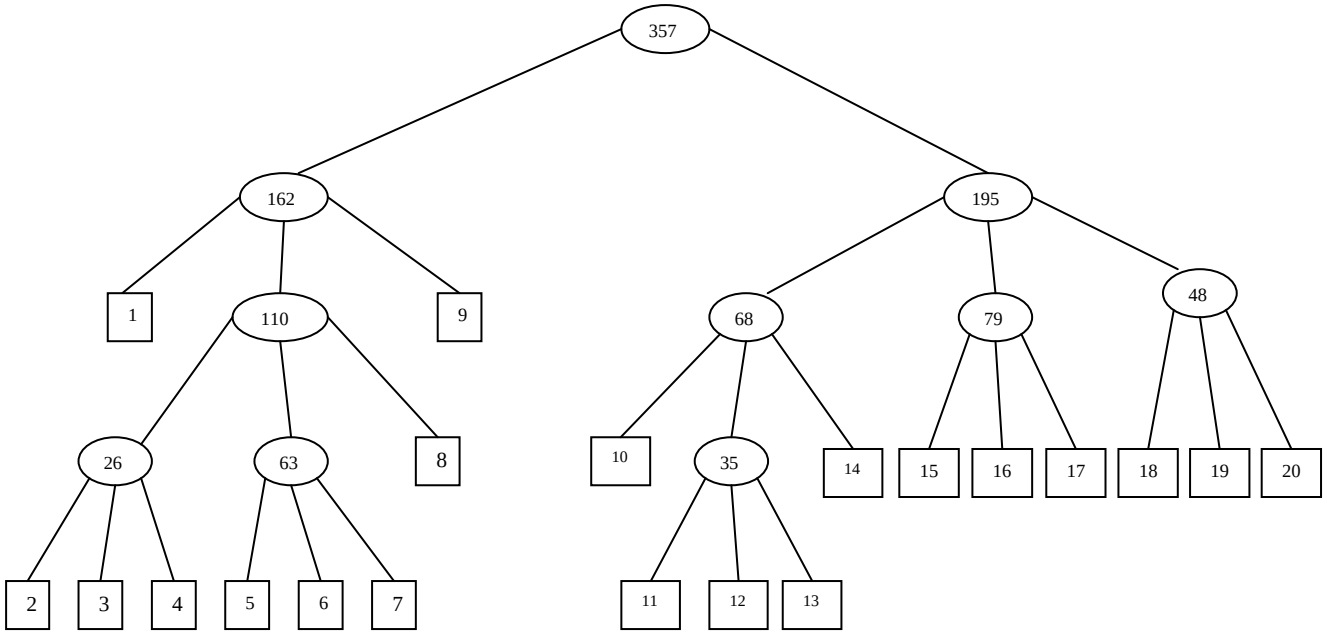

Fig 5 The k-ary Alphabetic Huffman Tree T''

.

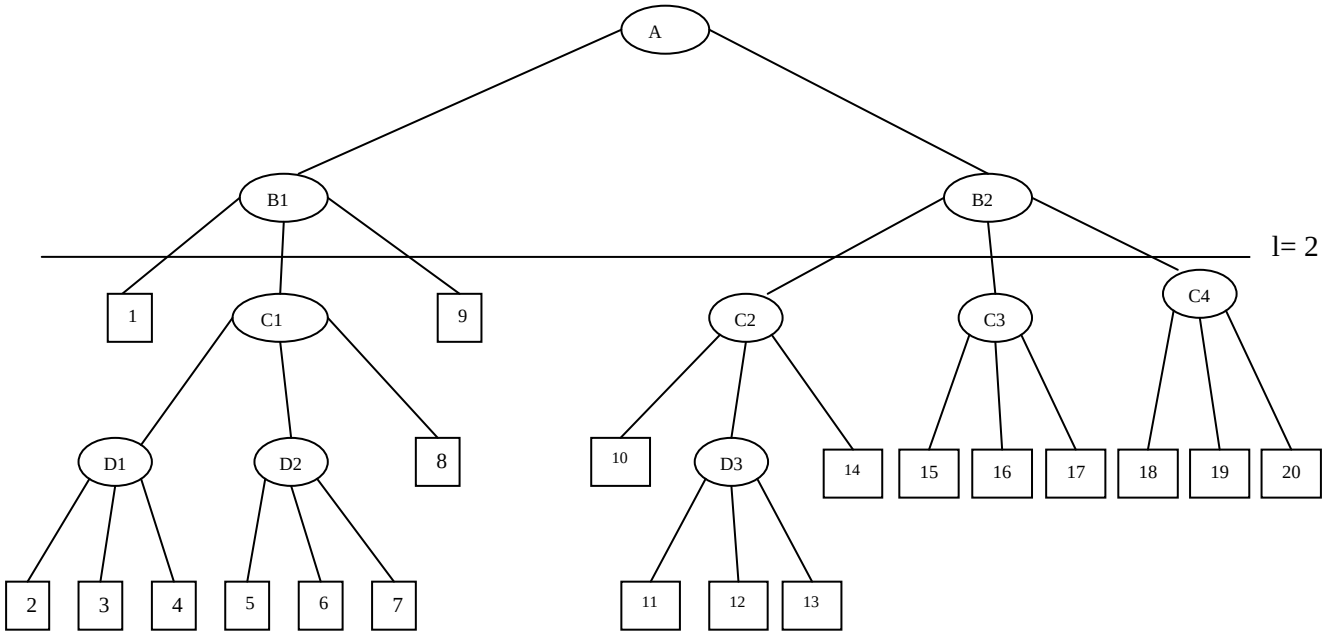

Fig 6 (a) The k-ary Alphabetic Huffman Index Tree T'''with replication at l=2.

$A^{[1]}$	$B1^{[1]}$	1	$B1^{[2]}$	C1	D1	2	3	4	D2	5	6	7	8	$B1^{[3]}$	9

$A^{[2]}$	$B2^{[1]}$	C2	10	D3	11	12	13	14	$B2^{[2]}$	C3	15	16	17

$B2^{[3]}$	C4	18	19	20

Fig 6(b) Broadcast sequence of k-ary Huffman tree T'''

4. ANALYSIS

The analysis between fixed fanout Huffman tree and varying fanout Huffman tree with replication is carried out. The fixed fanout Huffman tree is cut at fixed level *l*, we get control indices above *l* and rest are the search indices and the data items below *l*. The broadcast sequence of fixed fanout Huffman Tree Fig.4 is generated in Fig .7. The control and the search indices are highlighted by shaded grey. Since in this broadcast sequence every control index node is replicated for fixed number of times (for example two times). The index node A is traversed first, then B1 and B2 is traversed. Since the root node A and its child nodes B1 and B2 are replicated, so they will appear twice in the broadcast sequence, or we can say for fixed number of times. After seeing the Broadcast sequence of varying fanout index tree in Fig. (b) we found that root node A is replicated two times as in broadcast sequence of fixed fanout. But the index node B1 of Fig. 7 is replicated two and that of Fig. 6(a) is three times. Likewise happen with index node B2 would replicate two times in fixed fanout index tree but three times in varying fanout index tree. Hence it is possible in varying fanout index tree the index nodes replication would vary at different levels of the tree. It has been found that the depth of fixed fanout index tree is deeper than the proposed varying fanout index tree. Now, by taking into account both the broadcast sequence, we found that the length of fixed fanout tree is larger than the varying fanout index tree.

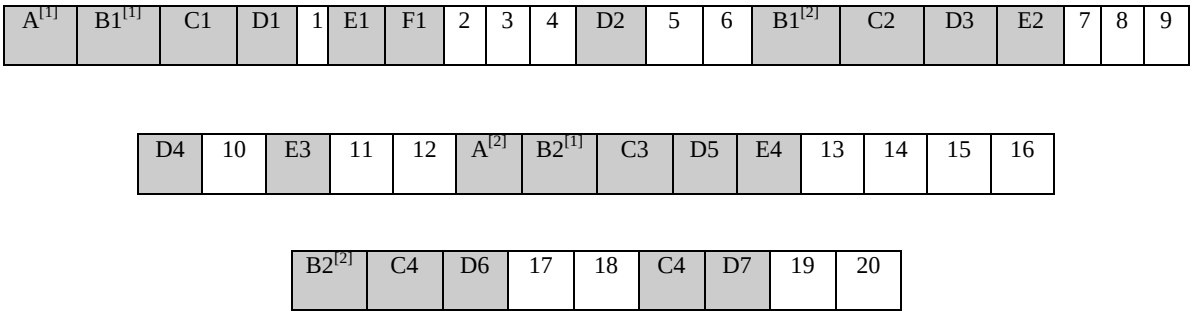

Fig. 7 Broadcast sequence of Huffman Tree T''

Table 3. Comparison between three index tree techniques

Issues	**Replicated Fixed Fanout**	**Replicated Varying Fanout (RVF)**	**Non replicated Varying Fanout**
Replicated index nodes	Fixed	Vary	Zero
Depth of index tree	Deeper	Less deeper	Same as RVF
Length of broadcast sequence	Larger	Smaller	Smallest
Directory miss	Decreased	Decreased	Increased

The theoretical analysis of replicated fixed fanout and varying fanout with non replicated varying fanout is provided in a Table 3.It has been found from comparison of replicated varying fanout (RVF) with replicated fixed fanout and non replicated varying fanout, that the proposed technique is a good solution for indexing with replication.

5. CONCLUSION

In this paper, we proposed a varying fanout indexing technique with replication for skewed data over a single wireless communication channel. Our proposed technique takes the frequency of each data item into consideration i.e. skewed pattern and accordingly generates the alphabetic Huffman tree. Then the replication is performed in the tree at any fixed level. The varying fanout tree is of smaller depth, which results a smaller broadcast sequence. The proposed indexing technique is compared and analyzed with the fixed fanout indexing technique for skewed data over a single wireless channel. The result shows the decrease in directory miss and the depth of the tree is also reduced.

The performance analysis of the proposed technique based on two parameters i.e. access time and tuning time is the future work.

6. REFERENCES

[1]T. Imielinski, S. Viswanathan, and B. R. Badrinath. "Energy efficient indexing on air". In Proceedings of the International Conference on SIGMOD, pages 25–36, 1994.

[2] A. Dan, D.M. Dias, and P.S. Yu, "The effect of Skewed Data Access on Buffer Hits and Data Contention in a Data Sharing Environment", Proc. 16th Large Dtabases Conf., pp.419-431, Aug.1990.

[3] Chen , M.S., Wu, K.L, " Optimizing index allocation for sequential data broadcast in wireless mobile computing", IEEE Transactions on Knowledge and Data Engineering, 15(1), pp. 161-173, 2003.

[4] Shivakumar, N., Venkatasbramanian, S.," Energy efficient indexing for Information Dissemination in wireless

Systems", ACM Journal of Wireless and Nomadic Application ,1996.

[5] Hu, T.C., Tucker, A.C.,"Optimal computer search trees and variable-length alphabetic codes" ,SIAMJ. Applied Mathematics, 21(1), pp. 514-532, 1971.

[6]Jiaofei Zhong ,Weili Wu and Yan Shi, " Energy Efficient Tree Based Indexing Schemes for Information Retrieval in Wireless Data Broadcast", DASFAA 2011, Part II, LNCS 6588, pp.335-351, 2011.

[7] S. Acharya, R. Alonso, M.J. Franklin, and S. Zdonik, "Broadcast Disks: Data Management for Asymmetric Communication Environments," Proc. ACM SIGMOD '95, pp. 199-210, May 1995.

[8]T. Imielinski, S. Viswanathan, and B.R. Badrinath, "Power Efficient Filtering of Data on Air," Proc. Fourth Int'l Conf. Extending Database Technology, pp. 245-258, Mar. 1994.

[9] W.-C. Lee and D.L. Lee, "Using Signature Techniques for Information Filtering in Wireless and Mobile Environments," Distributed and Parallel Databases, vol. 4, no. 3, pp. 205-227, July 1996.

[10]J. Xu, W.-C. Lee, and X. Tang, "Exponential Index: A Parameterized Distributed Indexing Scheme for Data on Air," Proc. ACM/ USENIX MobiSys, pp. 153-164, June 2004.

[11]J. Xu, W.-C. Lee, X. Tang, Q. Gao, and S. Li, "An Error-Resilient and Tunable Distributed Indexing Scheme for Wireless Data Broadcast," IEEE Trans. Knowledge and Data Eng., vol. 18, no. 3, pp. 92-404, Mar. 2006.

[12]J.Shen, Y.Chang," A skewed distributed indexing for skewed access patterns on the wireless broadcast", The Journal of Systems and Software 80(2007), Elsevier Inc., pp. 711-723, October,2006.

[13] J.Xu, DL Lee,Q Hu and WC Le, " Data Broadcast", Chapter11, Handbook of wireless networks and mobile computing, 2002, pp.243-265.

[14] Y. Yao, X. Tang, EP Lim and A. Sun," An energy efficient and access latency optimized indexing Scheme for Wireless Data broadcast", IEEE Trans. Knowledge and Data Eng.,vol. 18 no.8, pp. 1111-1124. August 2006.

[15] X.Xang and A. Bougettaya," Broadcast-Based Data Access in Wireless Environments", EDBT 2002, LNCS 2287, pp. 553-571, 2002.