行政院國家科學委員會專題研究計畫成果報告

第三代行動通訊網路中資料廣播之設計

Data broadcast on 3G mobile communication networks

計畫編號：NSC 90-2213-E-009-153
執行期限：90 年 8 月 1 日至 91 年 7 月 31 日

主持人：張明峰　　交通大學資工系
計畫參與人員：李隆盛、 楊偉儒、楊武桓　交通大學資工系

中文摘要

線廣播系統可很有效率將廣播資料送給一大群使用者，採用資料索引技術將可減少手機與基地台的聯繫次數(tuning time)進而降低電池電力消耗，索引的結構會影響資料索引的效率，我們提出演算法來組織索引並可減少手機聯繫基地台的次數，進而節省手機的使用電量。我們更提出一個評估原來手機聯繫基地台次數的效能量測參數來分析我們提出的方法。分析結果建議當資料的評存取機率的分配趨向均一時，索引結構中的索引連結數最好是 3 可以得到較低的手機聯繫成本。若資料的存取機率分布越不平均，即機率分配越偏斜，則索引連結數則最好是採用 2 比較可得到較低的手機聯繫成本。

關鍵詞：資料廣播，存取時間，接收時間。

Abstract

Data broadcast is an effective approach to disseminate information to a massive number of users. Indexing techniques for broadcasting data can reduce the battery power consumptions of mobile terminals by decreasing the tuning time. The organization of the indexes affects the efficiency of searching data. We present algorithms organizing the indexes to minimize the bandwidth required for the broadcast indexes, and thus minimize the tuning time. The analysis results show that the proposed algorithms outperform the Hu-Tucker algorithm. The numerical results also suggest the number of the branches of the index be 3 when the access probabilities of the data tend to be uniformly distributed so that the expected tuning time is minimal. When the access distribution of the data items is skewer, the tuning time can be reduced by setting the number of the branches in the index items 2.

Keywords: Data broadcast, tuning time, access time.

1. Introduction

DATA broadcast is an efficient technology to overcome the limited bandwidth. Data broadcast over radio channels allows users to access data simultaneously at a cost independent of the number of users. It is a powerful way to disseminate data to a massive number of users. A centralized server periodically broadcasts the data to a large number of mobile terminals through radio channels. The mobile terminals receive the broadcasts and filter out the data that is not desired. To evaluate the efficiency of the wireless broadcasting, two criteria are often used: access time and tuning time [1]. The access time is the average time interval from the moment a client requests data to the point when the requested data are received by the client. The access time determines the response time of data access. The tuning time is the time spent by a client listening to the channel. The tuning time determines the power consumed by the client to retrieve the requested data. Indexing techniques insert auxiliary information indicating when each data item will be broadcasted. After receiving the index, a client waits for the requested data most of time in the doze mode in which low power is consumed and only wakes up to receive data when the requested data is coming. Therefore, the tuning time can be reduced and the battery power is conserved.

In this paper, we proposed k-ary alphabetic index tree algorithms, as well as a precise tuning cost to evaluate the performance. The improvement in tuning cost is presented in this paper. The rest of the paper is organized as follows. In section 2, the system architecture is described. Section 3 presents the alphabetic Huffman algorithms. Section 4 shows the numerical results. Section 5 concludes this paper.

2. The System Architecture

To provide efficient search of the requested data, an alphabetic Huffman tree is used for the index tree. The clients using this scheme should first tune to the root of the index tree and traverse the tree to obtain the offset to the requested data.

Let n be the number of data nodes and the access probabilities of data nodes i be f_i. If the tuning time for data node i is T_i, the average tuning time can be expressed as $\sum_{i=1}^{n} f_i \times T_i$. The tuning time T_i can be determined by the depth of the data node in the index tree. Hu and Tucker proposed an algorithm to optimize the alphabetic-ordered Huffman code. Shivakumar and Venkatasubramanian suggested a k-ary Hu-Tucker algorithm to minimize the average tuning time, but didn't describe the algorithm clearly [2].

There is an open problem remained unsolved in the k-ary Hu-Tucker algorithm. The k-ary Hu-Tucker algorithm described by Shivakumar and Venkatasubramanian claims to construct the index tree with 2 to k branches, but didn't specify exact rules or algorithms to construct the tree [2]. A strategy to determine the degree of the internal nodes of an index tree to obtain the minimal average tuning time is needed.

If we increase the degree of an index, the depth of the index tree is decreased and the tuning time seems to be reduced. However, increasing the branches would increase the size of the index. For the wireless broadcasting, the indexes can be broadcasted on the index channels. The size of the index represents the bandwidth requirement of the radio channel. In wireless communications, a radio channel is partitioned into slots of constant size. Therefore, the size of the indexes should be the same to fit in a time slot. The tuning time should count the number of indexes received and the size of the index. Assume the depth of the data node i is d_i, and the degree of the index is k. β represents the length of the key and the offset. The average *tuning time* can be expressed as:

$$\overline{T} = \frac{k\beta \sum_{i=1}^{n} (d_i - 1) f_i}{\sum_{i=1}^{n} f_i} = k\beta \sum_{i=1}^{n} (d_i - 1) f_i \qquad (1)$$

3. The k-ary Alphabetic Huffman Algorithms

In this section, we propose an algorithm to construct the k-ary alphabetic Huffman tree and minimize the average tuning time. For n data nodes, it may not be possible to construct an index tree where all indexes have exact k downward branches. That is, there should be empty branches for some indexes. We call this type of index *incomplete* index. In our proposed algorithm, we first put those empty links in one index of the index tree, i.e., there is only one incomplete index in the tree. Under this assumption, the number of the non-empty links of the incomplete index can be determined from the number of data nodes, n, and the degree, k. Let a % b represent the remainder of a/b, and k_1 be the number of the non-empty links of the incomplete index. k_1 can be expressed as:

$$k_1 = \begin{cases} b + (k-1), & \text{for } n \,\%\, (k-1) = b \text{ and } b = 0 \text{ or } 1 \\ b, & \text{for } n \,\%\, (k-1) = b \text{ and } b \neq 0,1 \end{cases} \qquad (2)$$

Using the techniques of the binary Hu-Tucker tree, we construct the k-ary index tree by merging k nodes with the least access probability into an index node of the index tree. The number of the non-empty links of the incomplete index is calculated first. It is because that we reduce the average tuning time by merging nodes with less access probability into an index in the lower level of the tree. This algorithm will be referred to as the k-ary Incomplete-index First Alphabetic Huffman Algorithm (IFAH). The algorithm is shown in the following.

The IFAH Algorithm

```
Step 1. Let $S=(N_1, N_2, \ldots, N_n)$, the ordered list consists of
        all data nodes sorted by search key in increasing
        order. $N_i=D_i$.
Step 2. Calculate the number of the non-empty links $k_1$ of
        the incomplete index using Equation 2.
Step 3. Construct the incomplete index node.
    ● Find $k_1$ consecutive nodes in $S$ whose sum of access
      probabilities is minimum.
    ● Replace the $k_1$ consecutive nodes with an index node
      in $S$. The access probability of the index node equals
      to the sum of the access probabilities of the replaced
      nodes.
Step 4. If $|S|=1$, then go to Step 7.
Step 5. Construct the $k$-degree index nodes.
    ● Find the $k$ "consecutive" nodes, but index nodes can
      be bypassed, in $S$ that have the minimum sum of
      access probabilities.
    ● Replace the $k$ "consecutive" nodes with a new index
      node in $S$.
Step 6. If $|S|=1$, then go to Step 7. Else go to Step 5.
Step 7. Determine the level of each data node in the index
        tree.
Step 8. Reconstruct the index tree according to the levels of
        the data nodes.
    ● Initialize the ordered list $S$ as in Step 1.
    ● Find $k_1$ consecutive data nodes whose levels are the
      same. The levels of $k_1$ consecutive data nodes must
      be the maximum among the remaining nodes.
    ● Combine the $k_1$ consecutive nodes to an index node.
      Replace the $k_1$ consecutive nodes with the index
      node in $S$.
    ● Find $k$ consecutive nodes whose levels are the same
      and the maximum among the remaining nodes, and
      combine the $k$ consecutive nodes at the highest level
      first. Then, the nodes on the next-to-highest level are
      combined.
    ● The process continues until there is only one node
      left and its level must be 0.
```

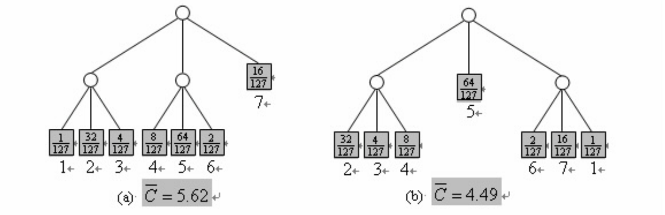

Fig. 1. The indexing trees with different alphabetic orders.

However, the alphabetic order of a k-ary alphabetic tree is not necessary fixed. Fig. 1 shows the index trees starting from different data nodes. The numbers on the links under the index nodes are the boundary key values of the index nodes. Fig. 1 (b) shows an example of *k*-ary alphabetic tree starting from D_2; the data node before the D_2 is rotated to the end of the ordered list. In the example of Fig. 1 (b), we show how to retrieve data node D_1. The boundary key values of the root index are 2, 5, and 6. Therefore, we chose the offset of boundary key value 6 to obtain the index of the next level because key value 1 for D_1 is less than 2. The alphabetic order of the data nodes in the index tree can be treated as a cycle. The average tuning times are 5.62 in Fig. 1 (a) and 4.49 in Fig. 1 (b), respectively. We apply can the rotatability to improve the IFAH. The new algorithm will be referred to as the *k*-ary Cyclic Incomplete-index First Alphabetic Huffman Algorithm (CIFAH). In Step 3 and 5 of IFAH, we scan from the left to the right of the ordered list to find consecutive nodes that have the minimum sum of the access probabilities. In CIFAH, we treat the ordered list as a cycle and find the minimum sum of access probabilities.

4. The Numerical Analysis

Uniform Access Probabilities

First, consider a special case where the access probabilities of the data nodes are identical, and the optimal index tree is a full *k*-ary tree that has no incomplete index. Let *d* be the depth of the index tree. The number of data nodes, $n = k^{d-1}$. The average tuning time, $\overline{T} = k(d-1)$. If *k* could be any real number, the average tuning time can be minimized.

$$\frac{d\overline{T}}{dk} = \frac{d(\frac{k \ln n}{\ln k})}{dk} = 0 \Rightarrow k = e = 2.71828...$$

Since *k* is a natural number, the result suggests that the average tuning time may be minimized when the degree of the index is 3.

Release the limitation of full *k*-ary tree, we consider the case that all data nodes are uniformly distributed. That is,

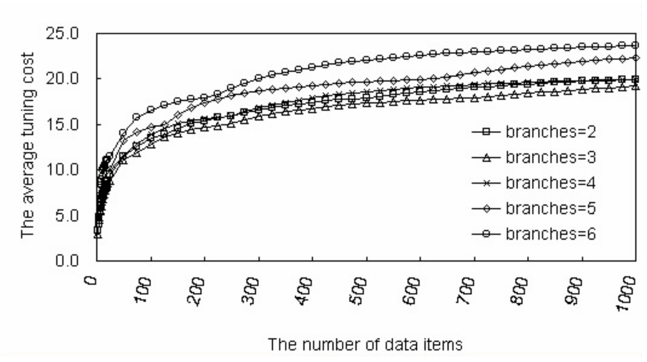

Fig. 2. The average tuning time for data with the uniform distribution.

$f_1 = f_2 = f_3 = \Lambda = f_n = 1/n$ and $d = \lceil \log_k n \rceil$.

The average tuning time can be expressed as:

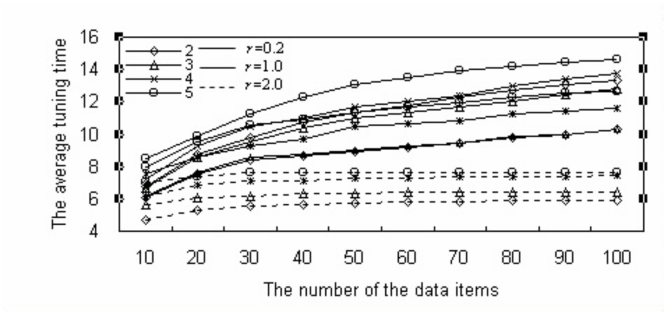

Fig. 3. The average tuning time of different number of the branches with *r*=0.2, 1.0, and 2.0 with the IFAH.

$$\overline{T} = \frac{k(n(d-1) - \left\lfloor \frac{k^d - n}{k-1} \right\rfloor)}{n} \tag{3}$$

Fig. 2 shows the tuning time as functions of the number of data nodes and the degree of the index node. The average tuning time increases as the number of data nodes increases due to the increasing height of the index tree. We focus on the optimal degree that leads to the minimal tuning time. The tuning time increases as the degree of the index is larger than 3. Therefore, when the access probabilities are uniformly distributed, the index nodes of degree 3 tend to minimize the average tuning time.

Zipfian Distribution in the Access Probabilities

Consider the case where the access probabilities are non-uniformly distributed. We assume the distribution of the access probabilities is Zipfian [3]. For *n* data nodes, the access probability of a data node D_i is as follows

$$f_i = \frac{1}{i^r \times \sum_{i=1}^{n} \frac{1}{i^r}}, \text{ where r is the rank of the distribution.}$$

Note that the larger the rank *r* is, the skewer the probability distribution is. In addition, f_i decreases as *i* increases. Fig. 3 shows the results of the average tuning time for different ranks of Zipfian distribution. For a small rank (e.g., *r*=0.2) and a large number of data nodes, the minimum average tuning time can be obtained when the degree is 3. It is because that a smaller rank for Zipfian distribution results in the less skew probabilities distribution. For a large rank (e.g., *r*=2) in Zipfian distribution, the minimal average tuning time is found when the degree is 2. This is because the large number of branches increases the tuning time of every data node in the

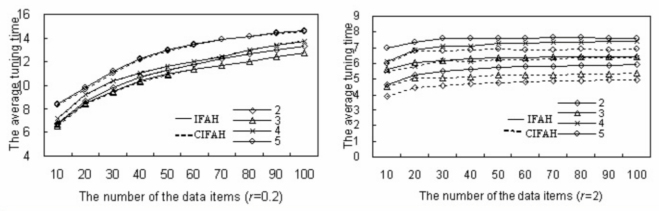

Fig. 4. The average tuning time of the IFAH and CIFAH.

index tree. Consider the index trees of a given degree. The skewer the access probability distribution is, the less the tuning time is. This is because as the access distribution gets skewer, fewer data nodes commands more access probability. The data nodes of large access probabilities trends to be placed at the lower levels of the index tree. As a result, the tuning time decreases.

Fig. 4 shows that the average tuning time of the CIFAH is less than that of the IFAH. It is because we can find the minimum tuning time from the selected nodes in the cycle sequence to build low cost indexes in the index tree. The improvement ratio of the CIFAH with large rank *r* is larger than that with small rank. This is because there are data nodes of larger access probabilities for the skewer access probabilities distributions. The CIFAH has the capability to find an ordered list for the data nodes to construct an index tree that places those frequently accessed data nodes in the lower level.

5. Conclusion

In this paper, we proposed indexing schemes to obtain minimal tuning time. The IFAH is an algorithm similar to the Hu-Tucker algorithm in organizing the indexes. To reduce the tuning time, the CIFAH can further improve the tuning time by rotating the sequence of the data nodes. The analysis results show that when the access probabilities of the data are uniformly distributed, the tuning time is minimal when the degree of the index node is 3. For the data nodes whose access probabilities are Zipfian distributed, the tuning time increases as the number of the data nodes increases. The CIFAH can effectively reduce the tuning time when the access probabilities are of Zipfian distribution, and the improvement becomes larger as rank *r* increases, i.e., the distribution gets more distorted

Reference

[1] T. Imielinski, S. Viswanathan, and B. R. Badrinath, “Energy Efficiency Indexing on Air,” In *Proceedings of the International Conference on SIGMOD,* pp. 25-36, 1994.

[2] N. Shivakumar and S. Venkatasubramanian, “Energy-Efficient Indexing For Information Dissemination In Wireless Systems,” *ACM-Baltzer Journal of Mobile Networks and Nomadic Applications*, vol.. 1, pp. 433-446, Dec. 1996.

[3] W. Li, “Random texts exhibit Zipf’s law-like word frequency distribution,” *IEEE Trans. Information Theory*, vol. 36, no. 6, pp. 1842, 1992.

附件：封面格式

行政院國家科學委員會補助專題研究計畫成果報告

※※※※※※※※※※※※※※※※※※※※※※※※※※
※　　　　　　　　　　　　　　　　　　　　　　　　※
※　　　　第三代行動通訊網路中資料廣播之設計　　　　※
※　　　　　　　　　　　　　　　　　　　　　　　　※
※※※※※※※※※※※※※※※※※※※※※※※※※※

計畫類別：■個別型計畫　　□整合型計畫
計畫編號：NSC　90-2213-E-009-153
執行期間：　90 年　8 月　1 日至　91 年　7 月 31 日

計畫主持人：張明峰　　交通大學資工系
共同主持人：
計畫參與人員：李隆盛、 楊偉儒、楊武桓

本成果報告包括以下應繳交之附件：
□赴國外出差或研習心得報告一份
□赴大陸地區出差或研習心得報告一份
□出席國際學術會議心得報告及發表之論文各一份
□國際合作研究計畫國外研究報告書一份

執行單位： 交通大學 資訊工程系

中　華　民　國　91 年　10 月　31 日